Chapter 2

An overview of social networks

A society is a collection of individuals which are interrelated. At its simplest form, a society may be represented by a network or graph. The graphs that arise this way are called *social networks*. Examining these networks can give us insight into how we interact with each other and the world around us, and what influences our behavior or success in different areas.

For instance, here are a couple of well-known examples of social networks.

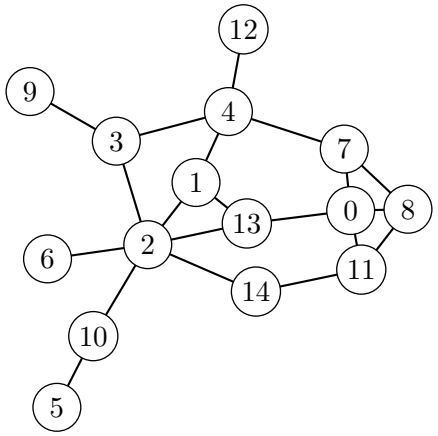


Figure 2.1: Florentine families ca. 1430

In Figure 2.1, each vertex represents a key Florentine family, and the edges denote a connection by marriage (Padgett and Ansell, 1993). One of these families rose to prominence and consolidated much of the power in 15th century Florence. Can you guess which one by looking at the graph? (Think about it now.)

Figure 2.2 depicts members of a Karate club, where the links represent friendship (Zachary, 1977). If I told you there was a schism in the club, and it later split into two rival clubs, can you guess how the split went just from looking at the graph? (Think about it now.)

While you may not have been able to give precise, definite answers to these questions, you probably guessed something pretty close to what actually happened. The fact that you can do this, even without any training, is a testament to how understanding the structure of networks can give you real insight into the workings of society. The Florentine family that rose to the top wasn't the one that was richest or most politically influential at first—its rise to prominence can instead explained by the social interrelationships of these 15 families.

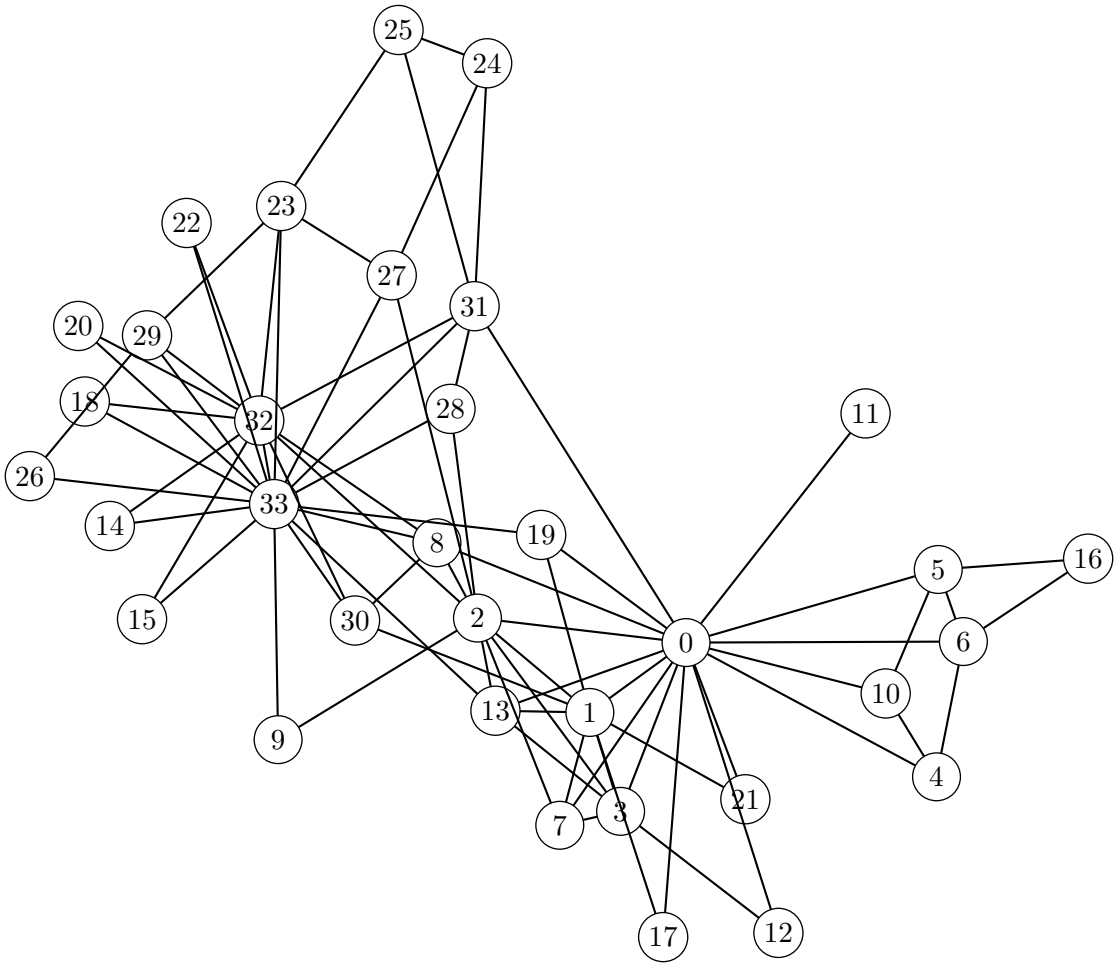


Figure 2.2: The Karate Club

Just like with classical graph theory, there are many points of view from which one can enter the study of social networks. One might come to social networks as a sociologist, or anthropologist, or linguist, or economist, or computer scientist, or engineer, or biologist, or businessperson, or investigator. Two principal questions a social scientist would ask are:

- What does the structure of a social network tell us about society?
- How do social networks form?

The first question can be taken at an local, individual level or a broader, global le Your position in society, who you're connected to, and who your connections are connected to, and so on, plays a large role in forming your beliefs, what opportunities you have, and your decision making.

Notice the second question, how do networks form?, implies that social networks change over time. They are dynamic. Vertices may come an go, edges may appear and disappear. Understanding how networks form will tell us how they may evolve and give us insight into questions such as how much do you your present connections play a role in forming future connections.

For good or for ill, I am not a social scientist, but I'll try to guide you into the area of social networks from the point of view of one mathematician. This means we'll ignore some questions

that are important to a social scientist (what is the best way to sample a network? how do you collect your data and turn it into a graph, e.g., if you're constructing a friendship network, how do you tell if two people are friends? how do you interpret graph invariants? etc), but focus on getting a deeper, structural understanding of networks. For us, two guiding questions will be

- How can we measure the structure of a network?
- How can we model social networks?

We've already seen some ways to get a handle on the structure of a network in the last chapter—e.g., connectedness, diameter, vertex degrees. Now we'll start to consider other measurable quantities of social networks that gives additional insight into the social structure of the network. For instance, we'll provide different ways to measure how important a given node is in a network. Oh, by the way, in the Florentine families network, the family that rose to power was the de Medicis (vertex 2). This was despite the Strozzis (vertex 4) being previously richer and more powerful. One graphic explanation is that vertex 2 has the highest degree, but a better explanation is that the de Medicis are more centrally positioned in the network. We'll look at some measures of *centrality* later.

Another aspect of this question is, we would like to characterize social networks in terms of important properties. We've seen that the graph isomorphism problem is difficult, which means we won't in general be able to determine a graph up to isomorphism by looking a few simple, easy-to-compute invariants. (Note: things like the adjacency matrices are not invariants, since they depend upon an ordering of vertices—there are up to $n!$ possible adjacency matrices for a given graph.) So instead, we'll examine what are some of the more important features/invariants of a graph from the point of view of social networks. This is particularly important when we are dealing with *complex networks*, i.e., really big networks like the internet.

This notion of characterizing social networks by key properties is important when it comes to the question of modeling social networks. By modeling social networks, we mean finding an algorithmic way to generate graphs whose key properties are essentially the same as those of social networks found in nature. These methods typically depend upon a random process—maybe at each stage you add/remove a node or edge with some given probability. Depending on these probabilities (which could be different for different nodes or edges), the kinds of networks you grow will look different. If these *random graphs* we generate have the same good-looking features as certain types of social networks, we'll consider these random graphs a model for these social networks. Note this also gives a model for how such social networks can form.

Over the past 20 years or so, there has be an explosion of interest in social networks, and there are lots of books on social networks now. Unfortunately, most of them are either not mathematical enough for our purposes or require a much more serious mathematical background. There are also recent introductory mathematics books about graph theory which discuss some aspects of social networks, but not everything we want to talk about. Two references I can recommend are *Social and Economic Networks* by Matthew O. Jackson (2008), an economist at Stanford, and *Networks, Crowds and Markets* by David Easley and Jon Kleinberg, an economist and computer scientist at Cornell (2010). The latter book is available freely online in preprint form at: http://www.cs.cornell.edu/home/kleinber/networks-book

In the rest of this chapter, we'll explain what are some "landmarks" or key features/invariants of social networks, and discuss a couple of basic ideas in network modelling, before delving more

deeply into some of these topics in our remaining two chapters. But first we'll talk about working with graphs in Sage.

Graphs in Sage

From now on, we'll do most of our computational work with graphs in the (free, open-source) mathematical package Sage, that has many built-in functions for working with and visualizing graphs. At present, the current version is Sage 6.1.1, though when I started writing these notes it was Sage 5.12. Most, if not all, of the things we will do should work on either of these versions of Sage, as well as any versions of Sage released in the near future. Sage is written in Python 2, and all of the things we've done in Python will also work in Sage.

(You may now wonder why we didn't start with Sage, and there are a couple of reasons. One, I think it is pedagogically better to start with Python first. I feel you get a better understanding of graph theory algorithms and how they are implemented by having to write some without yourself without too many tools available. Two, once you can use Python, Sage is a cinch. In addition, it's a useful skill to be able to program, and Python is a widely-used programming language, whereas most non-math people don't know what Sage is. So if you can put Python skills on your resume, that can be a big plus when looking for jobs.)

Let me also mention that there is a Python package called `networkx` for working with graphs. For some things, Sage is better, while `networkx` may be better for other things. I think both are equally difficult to install for Windows, but on a Mac, Sage is easier to install. Sage has the additional benefit that one can run it online without installation (e.g., https://cloud.sagemath.com/).

In any case, we will use Sage in this course, and now briefly explain how to get started working with graphs in Sage. See the lab page http://www.math.ou.edu/~kmartin/graphs/lab5.html for some other details and more references. This lab page also includes the data for the Florentine families and karate club graphs, in the Python adjacency matrix form we have been using so far.

First, Sage has a built-in graph object type, so to use the graph theory features of Sage on a graph we represented in Python with an adjacency matrix A, we'll need to convert it to a Sage graph object. We can do this by converting A to a Sage matrix (Sage also has a built-in matrix type), and then using the Sage matrix to generate a graph. For instance

Sage 6.1
```
sage: A = [ [0, 1, 0], [1, 0, 1], [0, 1, 0] ]
sage: m = matrix(A)
sage: G = Graph(m)
sage: G
Graph on 3 vertices
sage: G.show()    # or G.plot()
```

Here the lines that begin with `sage:` are what you input into Sage, and the other lines are the Sage output.

The commands `show()` and `plot()` are two functions that will draw the graph (slightly differently) on the computer for us. (Note neither of these commands draw the graph in a canonical way, so if you try plotting the same graph multiple times, it will not look the same each time.) I will not typically include the plots in these notes—you should try playing around with these or similar examples in Sage yourself and see the plots on your computer. There is also a `plot3d()` command

that gives you a 3-d visualization of your graph. Go to the lab page now, and use Sage to plot the Florentine family and Karate club graphs.

There are also many built-in constructors for generating graphs in Sage. You can see a list by typing `graphs.` (note the period) followed by *tab*. For example, try the following

Sage 6.1
```
sage: C10 = graphs.CycleGraph(10)
sage: C10.show()
sage: K5 = graphs.CompleteGraph(5)
sage: K5.show()
```

Pretty much everything we have talked about so far is already implemented in Sage. Here is a list of some basic commands. They are all run in the form `G.`[command], like the `show()` and `plot()` commands.

- `am()` — returns a (Sage) adjacency matrix
- `vertices()` — returns the list of vertices
- `order()` — returns the order
- `edges()` — returns the list of edges
- `size()` — returns size
- `degree(u)` — returns the degree of u
- `average_degree()` — returns the average degree
- `connected_components()` — returns the connected components
- `distance(u,v)` — returns the distance from u to v
- `shortest_path(u,v)` — returns a shortest path from u to v
- `diameter()` — returns the diameter
- `average_distance()` — returns the average distance
- `vertex_connectivity()` — returns the vertex connectivity
- `edge_connectivity()` — returns the edge connectivity
- `chromatic_number()` — returns the chromatic number

2.1 Landmarks of Social Networks

Here we discuss some, but not all, prominent features of social networks.

2.1.1 Connectedness

We've presented 3 real examples of social networks—the OU Math collaboration graph in Figure 2, the Florentine families graph in Figure 2.1 and the Karate club graph in Figure 2.2. Of these, the first is not connected but the latter two are. Some social networks are connected, but there's no reason to expect this in general. E.g., consider a graph of Facebook users, who are connected by a link if they are friends. There will be small groups of people who are only friends with each other, and therefore constitute make a connected component.

However, empirically, social networks tend to have a *giant component*. We'll give a precise definition in the context of random graphs (there's not a good precise definition for a single graph), but you can get some idea of what we mean by looking at the OU Math collaboration again—there's one connected component which is much bigger than the other ones, and it comprises a significant percentage of the total number of nodes in the network.

Here's another example—the Web Data Commons Hyperlink Graph (`http://webdatacommons.org/hyperlinkgraph/`). This is a directed graph where the nodes are webpages, and there is an edge from page A to page B if there is a hyperlink from page A to page B. This has (at present) about 3.5 billion nodes and about 128 billion links. If we look just at connected components in the undirected sense, this has a giant component consisting of about 3.34 billion nodes (94%). If we look at strongly connected components, there is a giant component consisting of about 1.8 billion nodes ($\sim 51\%$).

Of course, one has to be careful in interpreting this because of how the data was collected. This graph does not represent all webpages at a given time (this may not even be a sensible notion, as many webpages are dynamic—e.g., Google search result pages). Instead this data was collected from crawling the web. You find a node at random to start at, and then follow its links to find other nodes. If you only do this once, you'll only stay inside the connected component of the node you start at, and won't necessarily visit all pages with a link to your initial node. So you do this many times and collect the results. Still, this procedure tends to give you a graph that may be "more connected" than just picking a whole bunch of nodes at random. Of course, another issue is how do you find random webpages without finding them as links from another webpage? There are some methods, but there is no way to find all webpages with equal probability without being omniscient. (You'll never find hidden webpages.) However, this graph gives a good sense of the part of the web that is in common use.

2.1.2 Degree distributions

Euler solved the Königsberg bridge problem just by looking at the degree of vertices in the graph. Similarly, we can learn quite a bit about a social network just by looking at the degree of each node. We could just look at a list of the degrees of each node, and this can be done in Sage with the `degree_sequence()` function. However, it is often more convenient to count the number of nodes of a given degree, which can be done in Sage with the `degree_histogram()` function. We can either look at straight vertex counts (as the `degree_histogram()` function does), or normalize by the total number of vertices. The latter is what is known as the degree distribution, and this normalization allows us to easily compare degree distributions for graphs with different numbers of vertices.

Denote by $\mathbb{Z}_{\geq 0}$ the set of nonnegative integers $\{0, 1, 2, 3, \ldots\}$.

Definition 2.1.1. *Let $G = (V, E)$ be an undirected (not necessarily simple) graph. The* **degree**

distribution *of G is the function $P : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ defined by $P(d)$ is the proportion of nodes in V of degree d.*

Note the degree distribution defines a *probability function* on $\mathbb{Z}_{\geq 0}$—it satisfies $P(d) \geq 0$ for all d and the probabilities

$$\sum_{d=0}^{\infty} P(d) = 1$$

sum to one. (We'll review some basic probability theory later when we really need it.) For a given graph G of order n, of course we'll have $P(d) = 0$ whenever $d > n$ so the above sum is really a finite sum, and we can consider P as a probability function on the finite set $\{0, 1, 2, \ldots, n\}$. However, it is theoretically convenient to consider P as a function on all nonnegative integers so that (i) we can compare degree distributions for graphs with arbitrary number of nodes, and (ii) we can model some degree distributions with continuous functions like $P(d) = c/d^2$, where c is an appropriate normalization constant to make the total probability 1. (More on how to interpret the latter type of distribution below.)

We will also want to define degree distributions for directed graphs, to be able to talk about the Web Data Commons Hyperlink Graph for instance. In this case there are two kinds of degrees we can look at for a node.

Definition 2.1.2. *Let $G = (V, E)$ be a directed graph, and $u \in V$. The* **in degree** *of u is $|\{v \in V : (v, u) \in E\}|$, i.e., the number of edges which point to (end at) u. The* **out degree** *of u is $|\{v \in V : (u, v) \in E\}|$, i.e., the number of edges going out of (starting from) u.*

Definition 2.1.3. *Let $G = (V, E)$ be an directed (not necessarily simple) graph. The* **in degree distribution** *of G is the function $P : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ defined by $P(d)$ is the proportion of nodes in V of in degree d. The* **out degree distribution** *of G is the function $P : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ defined by $P(d)$ is the proportion of nodes in V of out degree d.*

Again, the in and out degree distributions define probability functions.

Note if G is given as an adjacency matrix A, we can find the out degree by counting the number of 1's in the row corresponding to u, and the in degree by counting the number of 1's in the column corresponding to u. On the other hand, if G is given as an adjacency list, it is easy to get the out degree of u. However, one has to go through the whole adjacency list to get the in degree, which is more time consuming (still polynomial time, in fact $O(n^2)$, but this is nontrivial if n is on the order of 1 billion).

If one is working with a really large directed graph, such as the Web Data Commons Hyperlink Graph, it is not feasible to store the graph as an adjacency matrix, so one has to store it as an adjacency list. So for large directed graphs, in degree is harder to get at than out degree. This is apparent if one thinks about how one has to collect the data. Given a webpage A, it is easy to find the out degree—just count the number of links to different pages on page A. However, if we want the in degree, we need to find all webpages that link to A, which is not easy. Consequently, it's not such a simple task to see how "popular" an individual page is (thinking of popularity in the sense of how many webpages link to this page—on the other hand, the server that hosts the webpage keeps track of how often it is accessed, so popularity in the sense of being accessed a lot is relatively easy to keep track of).

Let's consider two extreme degree distributions to see what kinds of things they tell us about a network. For purposes of illustration, let's think about friendship networks, i.e., the vertices are people and we connect them with a link if they are friends.

At one extreme, we could have a "delta-type" distribution, e.g., $P(5) = 1$ and $P(d) = 0$ for $d \neq 5$. This says the probability that a vertex has degree 5 is 1, so the probability a vertex has any other degree is 0. Since there are only a finite number of vertices, this means ever vertex has degree 5, i.e., our friendship network is 5-regular, i.e., everyone has exactly 5 friends. Of course this is not very likely, but we could loosen things up and have a "bump" distribution where most people have exactly 5 friends, some have 4 or 6 friends, and a few have 3 or 7 friends.

At the other extreme, we could have a distribution that runs the gamut of all possible degrees. There are different ways to do this—you could try for a "flat" distribution of the form $P(d) = c$ for $d \leq k$ and $P(d) = P(0) = 0$ for $d > k$. (This is not possible for $c = 1/n$; see Exercise 2.1.1.) (We could also choose to allow $P(0) = c$ so that there are nodes with no friends.) This would mean there are the same number of really popular people (with maximum degree k) as the same number of really lonely people (with only 1 friend), which is also the same as the number of people of average popularity (say $k/2$ friends). However a distribution like this is quite unlikely for social network, so let's think of how else we could do this.

If we broader our thought process a bit, we can think about another scenarios where people range over a gamut of possibilities. One would be something like exam grades, or IQ scores, which are often modeled with a Bell curve, which is a *normal* (or *Gaussian*) *distribution*. However, this is basically the same as what I called a bump distribution above. Another type of distribution where people run a whole spectrum of possibilities is a wealth distribution. Think about the distribution of wealth in the US. There are a lot of poor people, quite a few middle class, and a very few Richie Riches. The wealth distribution is often modeled by a **power law distribution**. This is a distribution of the form

$$P(d) = cd^{-\alpha}, \qquad \alpha > 1.$$

(Note this expression is infinite for $d = 0$, so we will set $P(0) = 0$ so there are no isolated nodes.) Here c is a normalization constant to make the total probability 1. Namely, for this to be a probability function, we need that

$$\sum_{d=1}^{\infty} \frac{c}{d^\alpha} = c \sum_{d=1}^{\infty} \frac{1}{d^\alpha} = c\zeta(\alpha) = 1,$$

where $\zeta(s)$ denotes the Riemann zeta function

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = 1 + \frac{1}{2^s} + \frac{1}{3^s} + \cdots,$$

which converges for $s > 1$. (This is why we insist $\alpha > 1$.) Hence, the constant $c = \frac{1}{\zeta(\alpha)}$, and we can write our power law distribution in the form

$$P(d) = \frac{1}{\zeta(\alpha) d^\alpha}, \qquad \alpha > 1.$$

We remark that explicit formulas for $\zeta(\alpha)$ is known when α is an even integer, e.g.,

$$\zeta(2) = \frac{\pi^2}{6}, \quad \zeta(4) = \frac{\pi^4}{90}, \quad \zeta(6) = \frac{\pi^6}{945}, \quad \zeta(8) = \frac{\pi^8}{9450}, \quad \cdots$$

In practice, ones need to compute $\zeta(\alpha)$ numerically, which is not hard. (Of course, in practice, even when α is even, one also has to approximate π.)

A graph that follows a power law distribution is called a **power law graph**. Many authors also call this a *scale-free graph* (and the power law distribution a *scale-free distribution*), though for some people the term scale-free has a more specific usage. The power law distribution is scale free in the sense that if you plot the graph in the range (.1, 10) or (.001, 1000), the graph will look the same, i.e., zooming out by a factor of 100 does nothing to the visual appearance of the graph. Consequently, if we have graphs with 100 nodes, 10,000 nodes and 1 billion nodes all with scale-free degree distributions with the same parameter α, then the degree distributions for these graphs will follow exactly the same shape. Roughly, the refined notion of a scale-free graph is a graph that looks the same when you look at the whole thing or zoom in to just see a part of it, similar to the way a fractal behaves. Such a graph must have a scale-free (power law) degree distribution, but having a power law degree distribution does not guarantee fractal like properties for a graph. We will use the term scale-free graph in its refined sense, and say power law graph when we just mean some graph with a power law distribution.

Now it may seem perplexing that for all $d = 1, 2, 3, 4, \ldots$, the proportion of nodes of any degree d can be nonzero. For instance, if we have a graph G on 100 nodes that follows a power law degree distribution $P(d) = \zeta(2)^{-1}d^{-2}$, how is it possible that $P(100) = 6/(100\pi)^2 \approx 0.00016 > 0$? Unless we allow loops or multiedges, the maximum degree of a vertex is 99. In addition, for a given graph G, the proportion of nodes of degree d must be a rational number, whereas $P(d) = 6/(\pi d)^2$ is irrational for all d. The answer of course is that no individual (finite) graph will have a degree distribution which is exactly a power law distribution. Rather, a power law distribution will be a convenient mathematical approximation of an actual distribution. It is helpful to think of $P(d)$ as being the probability that a given node has degree d, so $100P(d)$ should be approximately the number of nodes of degree d. Since $100P(100) \approx 0.016$, which rounds down to zero, this says we probabilistically expect 0 nodes of degree 100 (and similarly for higher degrees), and this indeed is what we logically expect.

Put another way, we can think of a power law distribution as a *model* for the degree distributions for certain networks. Allowing $P(d) > 0$ for all $d > 0$ in fact confers an advantage on us—this model is valid for all n. Returning to our example with $\alpha = 2$, we saw that $P(100) \approx 0.00016$. This means if we have a growing network following this degree distribution, by the time $n = 10000$, we expect to see 1 or 2 nodes of degree 100.

Many people think social networks roughly follow power-law distributions, so one often asks, given a social network, how close is the degree distribution to a power law distribution? The easiest way to see this visually is to look at what is called a *log-log plot* of the degree distribution. Note that if we have a power law graph $y = cx^{-\alpha}$, taking logs of both sides gives

$$\log y = \log(cx^{-\alpha}) = \log c + \log x^{-\alpha} = \log c - \alpha \log x.$$

So if we have data (x_i, y_i) that we suspect follows a power law for some exponent α, the data $(\log x_i, \log y_i)$ must satisfy a linear relation, and we can determine α by looking at the slope. The plot of the data $(\log x_i, \log y_i)$ is called the log-log plot. See Figure 2.3. There is a standard statistical procedure known as simple linear regression which can give us an objective measure of how close the data $(\log x_i, \log y_i)$ is to being linear, and approximating the slope α, but we won't discuss this here.

The web page http://webdatacommons.org/hyperlinkgraph/topology.html has log-log plots of the distributions of in degrees and out degrees for the Web Data Commons Hyperlink Graph. If

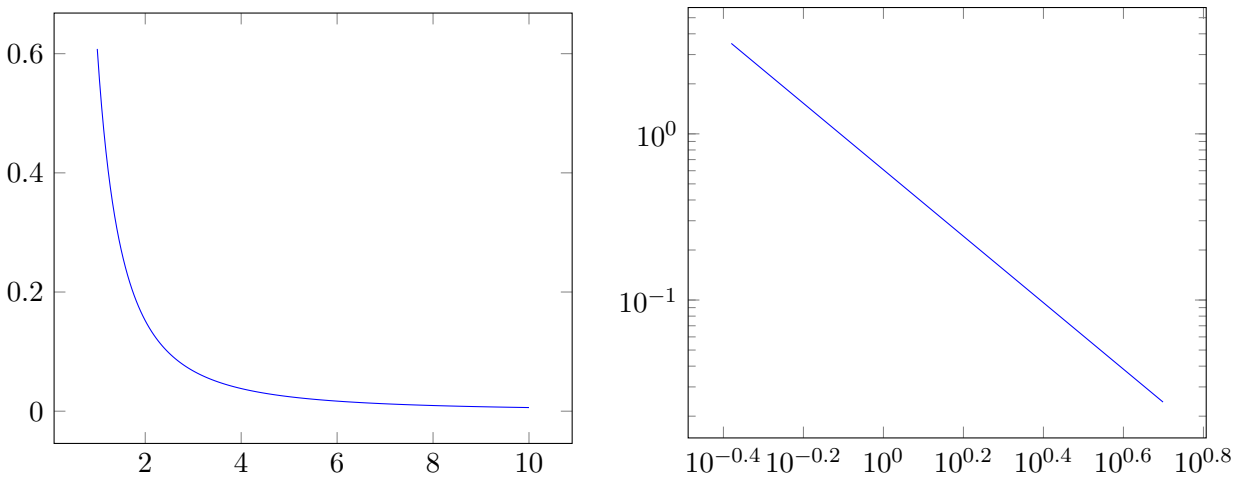


Figure 2.3: A power law distribution: the standard plot and a log-log plot for $\alpha = 2$

you take a look, you'll see the in degrees follow a power law distribution rather closely, at least for quite a while, but the out degree distribution does not look too linear in the log-log plot.

Let's take a look at the Karate Club graph from Figure 2.2. This has degree distribution given by $P(1) = P(9) = P(10) = P(12) = P(16) = P(17) = 1/34$, $P(2) = 11/34$, $P(3) = P(4) = 6/34$, $P(5) = 3/34$, $P(6) = 2/34$ and all other $P(d) = 0$. We've plotted the degree distribution in Figure 2.4 using both the standard and log-log plots. (Note, since $\log 0 = -\infty$, we need to omit the d's such that $P(d) = 0$ in the log-log plot.) Note the log-log plot does not look too linear, and indeed the standard plot does not look too much like a power function. One issue to take into account in this comparison is that the order of the graph is rather small ($n = 34$), so a few stray points on the graph will greatly alter its shape.

Looking at these graphs more closely, what are the differences with a power law graph? The first thing to notice is the Karate Club graph starts with a large spike, whereas a power law graph starts with a steep decline. If this friendship network really followed a power law graph, there should be a large number of people with only 1 friend, but here there's only one. Rather there are a large number of people with 2 friends, then the degree distribution drops sharply, and in the range $2 \leq d \leq 8$ (looking at the standard plot), it doesn't seem so far off from a power law shape. Indeed, the middle section of the log-log plot does not appear to be too far from linear. However, the last section (of both plots) go more-or-less horizontal. This is a function of working with a small value of n, so for the larger degrees where we don't expect too many such nodes, there are only really two possibilities that happen—for $d \geq 7$, either $P(d) = 0$ or $P(d) = 1$.

So what's our conclusion—is a power law a good model for this graph or not? Well, it's not my job to tell you how to think, so you can come up with your own conclusion. But we will revisit this question later when we talk about network models.

Here's how we can look at degree distributions is Sage.

Sage 6.1

```
sage: # K is the Karate club graph
sage: dh = K.degree_histogram()
sage: dh
[0, 1, 11, 6, 6, 3, 2, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1]
```

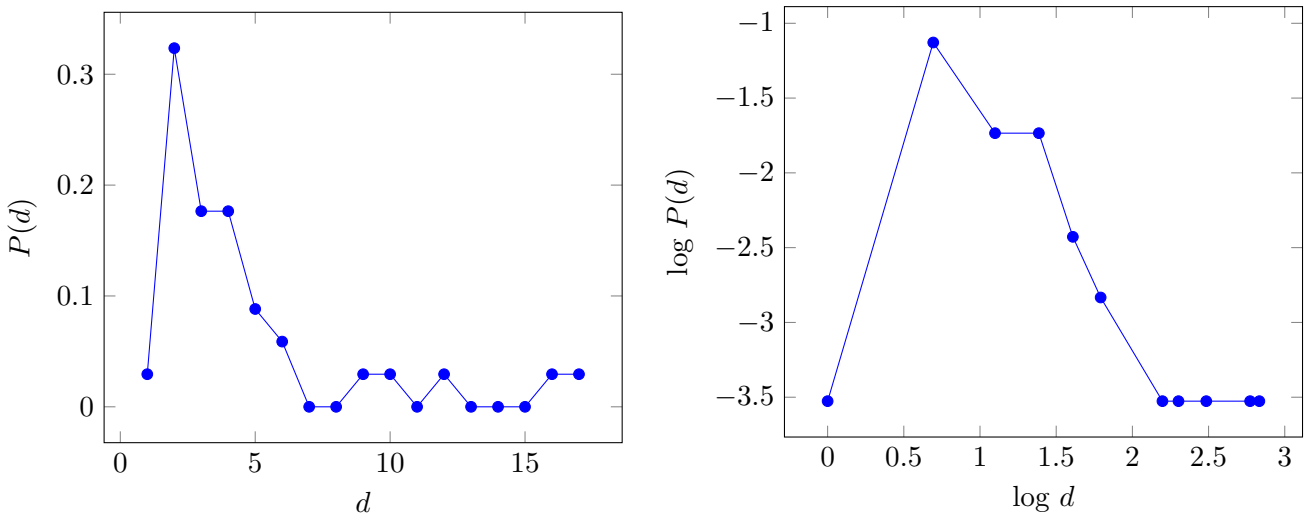


Figure 2.4: The Karate Club graph degree distribution — standard and log-log plots

```
sage: from sage.plot.bar_chart import BarChart
sage: bar_chart(dh)
```

There is no built-in function to get the degree distribution, but the `degree_histogram()` function returns a list whose d-th entry is the number of nodes of degree d (starting from $d = 0$). One can easily get the degree distribution from this, but for purposes of plotting, the degree histogram is sufficient. Sage supports line plots and bar chart plots—we chose to do a draw a bar chart for this example.

A related function is the `degree_sequence()` function, which returns a list consisting of the vertex degrees for each vertex, ordered reverse numerically.

2.1.3 Centrality

For our question about the Florentine families graph in Figure 2.1—which family rose to power—we want to understand how the "position" of a node in a network determines how much power or influence it has. There are different ways to try to measure this, and these notions fall under the heading of *centrality*.

The most basic measure of centrality is **degree centrality**. Given a (simple undirected) graph G, the degree centrality of a node is simply its degree divided by $n-1$. So if our node is connected to all other nodes, it has the maximum possible degree centrality 1, whereas if it is isolated, it has minimum possible degree centrality 0. In the Florentine families network, we see vertex 2 (the de Medici's, which is the family that rose to power) has the highest centrality, namely 6/13, or nearly 1/2.

However, degree centrality is a local notion, which I termed popularity in the Introduction. That is to say, the degree centrality of a vertex depends only on the neighbors of the vertex, and not how the rest of the network looks. According to (Padgett and Ansell, 1993), just looking at degree centrality in the Florentine families graph is not sufficient to explain how the de Medici's rose above the Strozzi's (vertex 4), who were richer and more politically powerful at the time.

In the Introduction, we observed in the collaboration graph in Figure 2 that while Przebinda and Özaydin have the same degree centrality, Przebinda is one step closer to most people in the large component. Similarly, the de Medici's are closer to more families than the Strozzi's in the Florentine graph.

To measure how close a node is to other nodes in a graph, we consider measures of **closeness centrality**. Then the inverse distance measure of closeness centrality is simply

$$\sum_{v \neq u} \frac{n-1}{d(u,v)}.$$

(For any v where $d(u,v) = \infty$, we regard $\frac{1}{d(u,v)} = 0$.) It is also natural to consider an inverse square measure:

$$\sum_{v \neq u} \frac{n-1}{d(u,v)^2}.$$

We normalized these measures by the maximum possible degree $n-1$ so if u has an edge to all other vertices v, these measures give the maximum value of 1. Of course one could consider arbitrary exponents $k \geq 1$ on $d(u,v)$ in these sums.

An alternative way to measure closeness centrality it to look at what is called *decay centrality*

$$\sum_{v \in V} \delta^{d(u,v)}, \qquad 0 < \delta < 1,$$

where δ is a parameter that can be adjusted. This does not require G to be connected as $\delta^\infty = 0$ for $0 < \delta < 1$. Note the limit as $\delta \to 1$ of the decay centrality of u is simply the size of the connected component of u.

Another notion of centrality is **betweenness centrality**. This measures the number of times a given vertex u lies on a (shortest length) path between other vertices v_1 and v_2, and gives a higher score the more times u appears. I won't give a precise definition, but both closeness and betweenness centralities give more "global" notions of centrality, better reflecting how well positioned a node is in a network. The de Medici's have high closeness and betweenness centralities, and this seems to be what gave them the ability to rise to power in 15th century Florence.

Sage has some of these centrality measures built in. Let's do an example.

Sage 6.1

```
sage: # F is the Florentine families graph
sage: # vertex 2 is de Medicis, vertex 4 is Strozzis
sage: F.centrality_degree(2)
0.42857142857142855
sage: F.centrality_degree(4)
0.2857142857142857
sage: F.centrality_closeness(2)
0.56
sage: F.centrality_closeness(4)
0.4666666666666667
sage: F.centrality_betweenness()
{0: 0.10256410256410256,
 1: 0.09157509157509157,
 2: 0.521978021978022,
 3: 0.21245421245421245,
```

```
4: 0.25457875457875456,
5: 0.0,
6: 0.0,
7: 0.1043956043956044,
8: 0.02197802197802198,
9: 0.0,
10: 0.14285714285714288,
11: 0.05494505494505495,
12: 0.0,
13: 0.11355311355311357,
14: 0.09340659340659341}
```

Here `centrality_degree()` is the normalized degree centrality, `centrality_closeness()` is the first measure of closeness centrality described above, and `centrality_betweenness()` returns the betweenness centrality of all nodes.

There is another, more sophisticated notion of centrality. How important a node is should be based on who it's neighbors are. The more important it's neighbors, the more important that node should be. This makes intuitive sense, but how can we use this idea to give a precise measure of centrality? We're basing our notion of importance on the notion of importance. There are various ways to resolve this recursive conundrum, the most elegant in my mind being **eigenvector centrality**. This notion forms the basis of the Google PageRank algorithm, and we will see how to define it in the next chapter with the notion of random walks.

2.1.4 Small world phenomena

Many social networks exhibit phenomena that are called *small world phenomena*. Loosely this means the network is very well connected, often surprisingly so. We already explored part of this idea a little bit with the notion of connectedness and giant components. Let's consider some large social network that you're part of—maybe the nodes are email or Skype or cell phone or Facebook users, and two people are connected if they've been in contact in the past month, or worse, are Facebook friends. For concreteness let's consider an actual (as opposed to Facebook) friendship network of students at OU.

Pretend you're in a story. Maybe you're really antisocial—you live off campus, you don't really meet other students, you just come to campus to go to class and leave, but on your way out from class, you randomly bump into this guy/girl who seems pretty cool, so you show him/her you're pretty cool too by not talking to him/her. Afterwards you realize this was pretty stupid, since you have no way of meeting him/her again. Still, you have this nutty professor for your Bokonism, Robotics and the African Diaspora class who makes you work in group projects, so you got together with this guy Kilgore in your group once or twice, and you guess you're sort of friends. He's also pretty antisocial, but even he has a couple other friends, and one of them has a bunch of friends in the Science Fiction Club. Some of the Sci-Fi'ers have friends in the Math Club. Everyone in the Math Club is friends, including that guy/girl you bumped into one and sort of liked, from whom you're only 4 steps away now. Gradually you meet Kilgore's friends, and eventually you start hanging out with some of the Sci-Fi kids. Then one of them who's in the Math Club tells you about this cool lecture he heard in the Math Club, and the good pizza. So you decide to check out the Math Club, and the next Wednesday you go up the the 11th floor of PHSC at 4pm only to discover no one's there. That's cause they don't meet every week. All of a sudden, the roof flies

off the building and you get sucked up in a tornado. Across from you, you see the guy/girl you kind of liked, and you work up the courage to wave and say hi. Only he/she doesn't wave back, because he/she is screaming uncontrollably, because the lower half of his/her is being eaten by a shark. But then the shark starts screaming because the lower half of his/her body (it's hard to tell the gender of a shark, especially in a tornado) is being eaten by a dinosaur. "All's well that ends well!" you scream. The end. True story. I have the screenplay rights.

The point is, even in you hang out with just a small group of people, though some random connections, you're connected to larger groups, which are connected to other large groups, and you'll see that most of the network is connected, or at least being eaten by sharks and dinosaurs. Sure, there may be some really antisocial people who don't make any connections, or little groups which are not connected to the rest of the network, but for friendship networks like this, we expect most people in the network to be connected to each other through some sequence of friends.

Another type of small world phenomena is what is usually referred to as *six degrees of separation*. In 1929, Frigyes Karinthy wrote a story (if you can call something without a sharknado a "story") about how the world was shrinking (not literally, unfortunately). In there, a character bets that if you choose any two humans on Earth at random, they are connected by a sequence of at most 5 personal acquaintances (not including themselves, so distance at most 6 in the "personal acquaintance network"). You could interpret this as a conjecture that the personal acquaintance network is connected with diameter at most 6, but this seems not likely to be strictly true, as there are pockets of societies with little-to-no contact with the rest of the world. However, a better interpretation is that if you choose two people at random, they are unlikely to be of distance more than 6 apart. I don't know if this is true or not—there's nothing inherently magical about the number 6—it was just some number Karinthy pulled out of his, um, writing cap—but I expect it is true for some relatively small distance. Incidentally, he phrase "six degrees of separation" was made famous by a play of the same name written by John Guare, but in his usage it means everyone is distance at most 7 apart.

Here is a heuristic reason. Let's make a very conservative estimate that you have made 100 acquaintances in your couple of decades on this earth. Similarly, your acquaintances will have at least 100 acquaintances each. So this yields potentially $100^2 = 10,000$ people of distance at most 2 from you. Of course there will be a lot of repetition in your acquaintances' acquaintances, but it seems reasonable to assume there are at least 1,000 people distance at most 2 from you. Continuing this form of estimation, you might estimate there are at least 10,000,000 people of distance at most 6 from you. Only a few more steps, and you're at 10 billion! See, you've got everyone covered! (This heuristic follows the same reasoning as the proof of Proposition 1.6.18.)

This seems like a hard problem to actually study at the level of personal acquaintances (how do you determine them, and how do you gather data?), but some studies have been done. One famous study was by Stanley Milgram in the 1960's who estimated the average distance between someone in Omaha and a specific person in Boston is less than 6. For this study, Milgram randomly chose 96 people in Omaha from the phone book, and sent them small packages, with instructions that they should send them to someone they know whom they think will be closer to the "target." That person should should do the same, recording the steps of the package. The target was a friend of Milgram's and specified by name, address and occupation. Of these packages, 18 reached Milgram's friend, with average path length about 5.9.

Another famous "study" is *six degrees of Kevin Bacon*. In 1994, Kevin Bacon was the center of the entertainment world (he said so himself!). Here we take the graph of all actors, and put

an edge between two if they worked on a film together. Any well-known actor will be in the same component as Kevin Bacon, which will be the giant component. At the time, there was a game to find a shortest path to Kevin Bacon, and usually the distance was less than six. Note this does not mean most pairs of actors are distance at most six away, though this may also be true—it only formally implies the distance between two random actors is usually less than 12.

Mathematicians have something similar, which is the notion of an Erdös number, and this is quite well-known too. Paul Erdös (1913–1996) was an itinerant Hungarian mathematician who wrote about 1500 papers with 511 different collaborators (though he almost never wrote anything up himself). Here we consider the collaboration graph of all mathematicians—two mathematicians are connected if they co-authored a paper together. The Erdös number of a mathematician is his or her distance from Erdös in the collaboration graph. Of course many mathematicians are not in the same component as Erdös, but of those that are (and there are many—it is a giant component), the average Erdös number is 4.65, the median is 5, almost all are below 8, and the maximum is 15. There are 511 with Erdös number 1 and 9,267 with Erdös number 2. (These numbers of course can change over time, and may be slightly dated now.) This provides some evidence that our estimate of 1,000 people of distance at most 2 from you in the acquaintance graph is reasonable (though Erdös is of course a special node), and that most people in a giant component are can be connected in a relatively small number of steps.

We can rephrase this mathematical notion of six degrees of separation as the statement: in the giant component of a social network, the average distance is relatively small and the diameter is is also not too large. We can be precise about the terms "relatively small" or "not too large" when we discuss random graphs.

Another aspect of small world phenomena is that given two nodes A and B, there tend to be many short paths from A to B. This is the feature that usually causes people to declaim, "It's a small world." For example, say there are two friends Abbey and Camille, and Camille introduces Abbey to her new ocarina teacher, Barry. Guess what? Abbey already knows Barry because he plays the card game Set with Abbey's brother Doug on Tuesday nights. Now there are two short paths which represent the ways Abbey knows Barry—through Camille, and through Doug. As they say, it's a small world.

This idea that there are often many short paths between two vertices is related to the notions of cliques and clusters, which we'll discuss next.

2.1.5 Cliques and clusters

A **clique** in an (undirected) graph G is a subgraph of G isomorphic to a complete graph. Cliques of order 1 are just single vertices. Cliques of order 2 are just pairs of vertices connected by an edge, so for simple undirected graphs the number of cliques of order 2 is just the number of edges. Cliques of order 3 are "triangles" in the graph, or just cycles of length 3 if we ignore the initial vertex (i.e., if we consider the cycles (a, b, c, a), (b, c, a, b) and (c, a, b, c) to be the same).

For instance, consider K_4. This has one clique of order 4 and 4 of order 3, 6 of order 2 and 4 of order 1. On the other hand, the cycle graph C_n for $n > 3$ has no cliques of order > 2.

Thinking in terms of a social network, say a friendship network, a clique is a subset of of people such that any two people in this set are friends. Knowing what the cliques are in a network will tell us a lot about the society. If you and I are friends, are most of your friends my friends also? Is the formation of the network very clique-ish (i.e., are most connections made by "networking"?), or are most connections made by chance encounters? In the first case, we expect many fairly large

cliques, where as in the second, not many at all.

The **clique number** of G is the maximum order of a clique in G. For K_n, it is clearly n. For C_n it is 2 unless $n = 3$, in which case it is 3 as $C_3 = K_3$. In the OU Math collaboration graph, the clique number is 3, but there are 7 cliques of order 3. For the Florentine families graph, it is also 3, with 3 cliques of order 3. For the karate club graph, the clique number is 5—this graph has 2 cliques of order 5 and 11 cliques of order 4 (2 of which are not contained in a clique of order 5).

Sage has a many built-in functions for cliques. Here is an example of a few.

Sage 6.1

```
sage: # Here F is the Florentine families graph
sage: F.clique_number()
3
sage: F.cliques_maximum()
[[0, 7, 8], [0, 8, 11], [1, 2, 13]]
sage: F.cliques_maximal()
[[0, 8, 7],
 [0, 8, 11],
 [0, 13],
 [2, 1, 13],
 [2, 3],
 [2, 6],
 [2, 10],
 [2, 14],
 [4, 1],
 [4, 3],
 [4, 7],
 [4, 12],
 [5, 10],
 [9, 3],
 [11, 14]]
```

As you can guess, `clique_number()` returns the clique number of the graphs. The function `cliques_maximum()` returns all the cliques of maximum possible order. The function `cliques_maximal()` returns all *maximal cliques*, meaning they are not contained in larger cliques.

One issue with counting cliques in a social network is that cliques are unstable under minor changes. Remember that social networks are dynamic, so we want to look at measures of graphs which are robust in the sense that they do not vary wildly under small changes to the network. For instance, just adding the single edge $\{7, 11\}$ would change the clique number of the Florentine families graph from 3 to 4.

Even from a "static" point of view, the notion of clique is rather rigid for what we want to measure. We might want to look for "generalized cliques" in the network—tightly-knit groups in the network where perhaps not every pair in this network is directly connected, but most are. This brings us to the notion of *cohesiveness* and graph decompositions.

Let's go back to the example of the karate club in Figure 2.2. The karate club graph has 2 "hubs", vertex 33 of degree 17, and vertex 0 of degree 16. One is the instructor of the club and the other is the student founder. From a networks point of view, this graph can be roughly partitioned into 2 subgraphs around these hubs which are cohesive (relatively tightly-knit) groups. Here is a picture of how the split went.

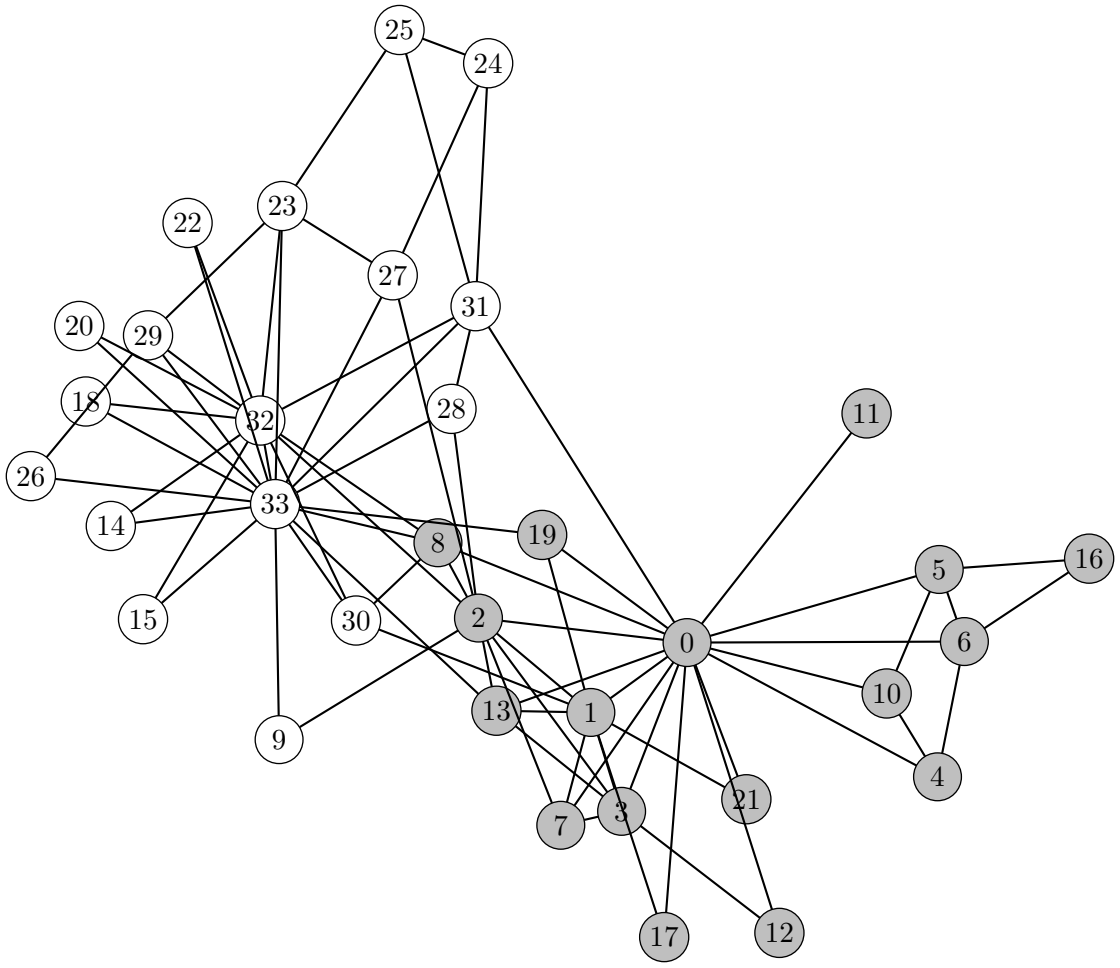


Figure 2.5: The Karate Club, split

For the most part, the split can be determined in the following way—*karateka* i went with vertex 0 or 33, according to whoever they were closer to. However, this does not provide a complete explanation. For instance, 19 is friends with both 0 and 33, but chose to stay with 0. Why? Well, a likely explanation is that 19 is also friends with 1, who is friends with 0 but not 33. When faced with a choice, people often do what their friends do. This reasoning does not explain all choices made—e.g., 8 went with 0 despite having more friends who went with 33. We don't capture all relevant information in this network (e.g., strength of connections and any personal details are not measured), but the point is we can mathematically guess how the split would go almost exactly even before it happened by just looking at the network structure.

Here is one way of mathematically formulating the idea that friends tend to go with friends' choices. We partition the graph into two components, one containing 0 and one containing 33, in such a way that we minimize the number of triangles (cliques of order 3) broken. For example, if 19 went with 33, we would loose the clique $\{0, 1, 19\}$, but if 19 goes with 0, no clique of order 3 is destroyed.

How can we identify cohesive subgraphs? While cliques are certainly cohesive, this measure is too crude to identify the two components the karate club split into. One strategy is to use the

notion of *clustering*. Visually this is the notion of how many triangles are in the graph and how much they bunch together. There are different ways we can measure clustering.

Definition 2.1.4. *Let $G = (V, E)$ be a simple undirected graph. The* **(individual) clustering coefficient** *of a node $u \in V$ is the probability $Cl(u)$ that two neighbors of u are themselves neighbors, i.e.,*

$$\begin{aligned} Cl(u) &= \frac{|\{(v,w) : v, w \in V; v \neq w; (u,v), (u,w), (v,w) \in E\}|}{|\{(v,w) : v, w \in V; v \neq w; (u,v), (u,w) \in E\}|} \\ &= \frac{|\{(v,w) : v, w \in V, v \neq w; (u,v), (u,w), (v,w) \in E\}|}{\deg(u)(\deg(u)-1)}. \end{aligned}$$

The **average clustering coefficient** *is the average of $Cl(u)$ over all $u \in V$. The* **overall clustering coefficient** *or* **transitivity** *of G is*

$$Cl(G) = \frac{\sum_{u \in V} |\{(v,w) : v, w \in V; v \neq w; (u,v), (u,w), (v,w) \in E\}|}{\sum_{u \in V} |\{(v,w) : v, w \in V; v \neq w; (u,v), (u,w) \in E\}|}.$$

The meaning of the individual and average clustering coefficients are straightforward—e.g., for friendship networks this measures how many of your friends are friends with each other. We'll explain transitivity momentarily.

First let's look at some calculations for the karate club graph `K`.

Sage 6.1

```
sage: K.clustering_coeff([0,33])
{0: 0.15, 33: 0.11029411764705882}
sage: K.cluster_triangles([0, 33])
{0: 18, 33: 15}
sage: K.clustering_average()
0.5706384782076823
sage: K.cluster_transitivity()
0.2556818181818182
```

The function `clustering_coeff` returns the individual clustering coefficients for the list of nodes specified (or all nodes by default). The function `cluster_triangles` returns the number of triangles involving node u for each u in the specified list (or all nodes by default). The `clustering_average()` and `cluster_transitivity()` functions return the average and overall clustering coefficients for the graph.

Note that even though the vertices 0 and 33 are involved in more triangles than any other vertices, their individual clustering coefficients are much lower than the average clustering coefficient. This is actually to be expected with hubs, because they are friends with so many different people. On the other hand, vertices like 18 or 20, who are only friends with 32 and 33 (who are friends), have clustering coefficients 1. When we take an average of individual clustering coefficients, the clustering coefficients for all vertices are weighted the same.

If instead we want to put more weight on vertices with higher degree, we can see this in the overall clustering coefficient. We call a "potential triangle" in a graph a *triad*, i.e., a triad is a set of 3 vertices in the graph with at least 2 edges among them. Then the overall clustering, or transitivity, as defined above is simply the number of triangles in the graph divided by the number of triads.

Both the average and overall clustering coefficients give some sense of how cohesive the graph is. Let's examine these clustering measures for the 2 graphs `K1` and `K2` of how the karate club split. Here `K1` is the subgraph of `K` consisting of people that went with 0, and `K2` is the subgraph of `K` consisting of people that went with 33.

Sage 6.1

```
sage: K1.cluster_transitivity()
0.39195979899497485
sage: K1.clustering_average()
0.7215686274509804
sage: K2.cluster_transitivity()
0.25139664804469275
sage: K2.clustering_average()
0.631279178338002
```

In both cases we see there is a sizable jump in the average clustering coefficients over the original graph `K`. There is also a significant jump in overall clustering for `K1`, but almost no change (in fact a slight drop) for `K2`.

How does this compare with clustering coefficients for random subsets of `K`? I generated a couple of random subsets of vertices of size 17, and for both of these the overall clustering was not too far from 0.25, but the average clustering dropped to about 0.43.

This suggests that clustering coefficients can provide useful measures of how cohesive certain groups in a network are. Clustering will also give us some insight into how a network forms. A lot of clustering in, say, a friendship network suggests that many friendships were formed through mutual friends. On the other hand, in networks with relatively low clustering suggests that most connections were formed not through mutual connections, but other methods, such as chance encounters or strategic choices (e.g., in the Florentine families network, it is likely that many of the connections (marriages) were strategically formed for mutual gain).

Exercises

Exercise 2.1.1. *Fix $k \in \mathbb{N}$. Show there is no simple undirected graph G with exactly 1 vertex of each degree $1, 2, \ldots, k$ and no vertices of higher degree. (Hint: what can you say about the number of vertices of degree > 0?)*

Exercise 2.1.2. *Is there a simple undirected graph G with exactly 2 vertices of each degree $1, 2, 3, 4$ and no vertices of higher degree?*

Exercise 2.1.3. *Write a Sage function* `degree_distribution(G)` *that takes in Sage graph* `G`*, and returns the degree distribution. This should be returned as a list, just like* `degree_histogram()`*, where the d-th entry is the proportion of nodes of degree d (starting with $d = 0$. Test this on the Florentine families and karate club graphs. (You define functions in Sage just as in Python.)*

Exercise 2.1.4. *Write a Sage function* `loglogplot(dd)` *that takes in a degree distribution* `dd` *and produces a log-log plot. (Use a line plot—see the Sage documentation. The function* `log` *is built into Sage. You may also want to use the* `float` *function which converts numbers into floating point numbers.) Test this on the karate club graph and compare with Figure 2.4.*

Exercise 2.1.5. *Write a Sage function* `decay_centrality(G,u,delta)` *which returns the decay centrality for vertex* `u` *with parameter* `delta`*.*

Exercise 2.1.6. *Write a Sage function* `clique_count(G,m)` *that returns the total number of cliques of order* m *in* `G`*. Note you cannot read this information directly off the* `cliques_maximal()` *result as this only returns maximal cliques. Test your function on the Florentine families and karate club graphs.*

Exercise 2.1.7. *Design an algorithm to predict which nodes went with vertex 0 and which went with vertex 33 in the karate club split. Explain your algorithm, code it up in Sage, and compare your results to data in Figure 2.5.*

2.2 Social Network Models

To properly study social networks, it is important to have a notion of *social network models*. The basic idea is that we have some procedure, usually involving an element of randomness, that generates graphs. (We've seen one example with random tree generation in Exercise 1.6.7.) Typically there are some adjustable parameters in this procedure. If, for some choice of parameters, this procedure generates graphs which typically look very similar to (have the same landmarks as) a given social network G_0, this can give us a lot of insight into how the social network G_0 has formed, and how it will evolve over time.

In addition, models will allow us to formalize intuitive notions about networks and prove theorems about them, which will give us baseline expectations for our social networks. For instance, we'll be able to say what it precisely means to have a giant component and show that certain types of networks almost always have giant components. On the other hand, when a network looks different from our expectations, this will tell us that there is something interesting going on about how this social network forms.

2.2.1 Probability for n00bs*

Because we want to talk about randomness, we first need to establish some basic notions from probability.

Definition 2.2.1. *A* **(discrete) probability space** (S, P) *is a set* S *with a function* $P : S \to [0, 1]$ *such that*

$$\sum_{s \in S} P(s) = 1.$$

The function P *is called the* **(discrete) probability function***, or* **(discrete) probability distribution***.*

It is convenient to extend the definition of the probability function P to subsets $A \subset S$ by

$$P(A) = \sum_{s \in A} P(s).$$

Then it is clear that P satisfies the properties

$$0 \leq P(A) \leq 1, \quad A \subset S$$

*Certainly not for dummies, but smart people who haven't seen or don't remember this stuff.

(with $P(\emptyset) = 0$ and $P(S) = 1$) and

$$P(A \cup B) = P(A) + P(B), \quad A, B \subset S,\ A \cap B = \emptyset.$$

We will call subsets $A \subset S$ **events** in the probability space.

Once P is understood, we often just refer to S as the probability space. We can think of this probability space as follows. We have some infinitely repeatable experiment (under identical conditions), and the elements $s \in S$ represent the different possible mutually-exclusive outcomes of this experiment (exactly one outcome s in the space S occurs after each trial of the experiment). The probability $P(s)$ represents the fraction of the time we get outcome s. Similarly, the probability $P(A)$ represents the fraction of the time we get an outcome $s \in A$.

Example 2.2.2. *We can model a fair coin flip with a probability space $S = \{H, T\}$, where H represents heads and T tails, where the probability function is defined by $P(H) = P(T) = 1/2$.*

Example 2.2.3. *We can model a fair die roll with a probability space $S = \{1, 2, 3, 4, 5, 6\}$, where $P(s) = 1/6$ for all $s \in S$. Then the probability of rolling an even number is $P(\{2, 4, 6\}) = 1/6 + 1/6 + 1/6 = 1/2$. Here the set $A = \{2, 4, 6\}$ represents the event of an even die roll.*

This gives a little indication as to why it is convenient to define probabilities of subsets, rather than just elements, of S. So does the next example.

Example 2.2.4. *Consider an urn with n balls, m of which are red, and $n - m$ are green. You draw one at random (the balls are mixed so that each ball is drawn with equal probability). We can model this with a probability space as follows. Think of the balls as being numbered from 1 to n, and for convenience assume that balls 1 through m are red, and the remaining balls are green. Formally, take $S = \{1, 2, \ldots, n\}$ and $P(s) = \frac{1}{n}$ for each n. Then $A = \{1, 2, \ldots m\}$ represents the event of drawing a red ball and $B = \{m + 1, m + 2, \ldots n\}$ as represents the event of drawing a green ball. Hence the probability that you pick a red ball is $P(A) = P(1) + P(2) + \cdots + P(m) = \frac{m}{n}$, (unless you are red-green color blind, in which case the probability is 1).*

Alternatively, we can model this experiment with the probability space (S', P') where $S' = \{R, G\}$ and $P'(R) = m/n$ and $P'(G) = (n - m)/m$. However the first model is more convenient if one want to allow things like drawing k balls (without replacement) and counting how many are red or green.

Example 2.2.5. *Let $G = (V, E)$ be an undirected graph. Then (V, P) is a probability space where P is the degree distribution.*

All of the examples of probability spaces we just have have been finite spaces, but we do not need to restrict ourself to finite spaces. In fact, we've seen a non-finite discrete probability space before.

Example 2.2.6. *Fix $\alpha > 1$. Let $S = \mathbb{N} = \{1, 2, 3, \ldots\}$ and $P(s) = \frac{1}{\zeta(s)s^\alpha}$. Then (S, P) is a probability space, where P is the power law distribution described in Section 2.1.2.*

The other major type of probability space is a *continuous probability space*. (One can also have mixtures of continuous and discrete probability spaces.) For instance if $S \subset \mathbb{R}$ is an interval, then the distribution function will be an integrable function $f : S \to [0, \infty)$ such that $\int_S f(x)\,dx = 1$. Events will be subsets $A \subset S$ and the probability of an event is given by $P(A) = \int_A f(x)\,dx$.[†]

[†]Technically, A should be what is called a *measurable* subset of S (think of a union of intervals) since there exist strange sets that don't make sense to integrate over. This is also why I assumed S was an interval.

To be a little more concrete, think of $S = [0, 1]$, and our random process is picking a number between 0 and 1. If we model this with any continuous distribution f, then if $A = \{a\}$ is a single point, we see $P(A) = \int_a^a f(x)\,dx = 0$. In other words, the probability of picking any specific number must be 0, in contrast to the case of discrete distributions. This of course doesn't mean you never get any specific number—you always get some specific number, but you can think of this as follows: even if you pick a (countably) number of random numbers between 0 and 1, according to this distribution, you will never pick the same number twice.

Rather for continuous distributions, it is only meaningful to discuss the probability that our random number lies in a given range. The simplest possible example is if we take $f(x)$ to be the constant function $f(x) = 1$—this is called the **uniform distribution** (on $[0, 1]$). For the uniform distribution (on $[0, 1]$), we have $P([a, b]) = P((a, b)) = \int_a^b dx = b - a$, i.e., the probability our random number lies in any given interval is simply the length of that interval (and it doesn't matter whether we include endpoints or not, since the probability of an endpoint value is 0).

Python (and by extension Sage) has a built in (pseudo)random number generator called `random()`, which returns a random number between 0 and 1.

Python 2.7

```
>>> from random import random   # you don't need this line in Sage
>>> random()
0.22445061772144392
>>> random()
0.103016249389979
>>> random()
0.90772991525930202
```

Technically this is a pseudorandom number generator, not a random number generator, because the method is deterministic, i.e., not truly random. However, it is close enough to random for our purposes, in the sense that each time `random()` is called, the probability the result x lies in a specific range $0 \le a \le x \le b \le 1$ is effectively $b - a$. In particular, if we want to do something with probability p, say add 1 to some variable a with $p = 0.5$, we can write this in Python/Sage as follows:

Python 2.7

```
>>> a = 0
>>> p = 0.5
>>> if random() < p:
...     a = a + 1
...
>>> a
0
>>> if random() < p:
...     a = a + 1
...
>>> a
1
```

This does what we want, because the probability that `random() < p`, i.e., that `random()` lies in $[0, p]$ is $p - 0 = p$.

There is one other basic concept we need from probability, and that is the notion of *independent events*. To understand this concept, consider the following example. Suppose we have an urn with

3 red balls and 5 green balls, and draw 2 balls in sequence (without replacing the first one). There are two events that we are interested in—the color of the first ball and the color of the second ball. The probability that the first ball is red is clearly 3/8. What about the probability the second ball is red? It depends on whether the first ball was red or green—it is 2/7 if the first ball was red, and 3/7 if the first ball was green. Hence these events are not independent. Though it is still possible to calculate the absolute probability the second ball is red:

$$\frac{3}{8} \cdot \frac{2}{7} + \left(1 - \frac{3}{8}\right) \frac{3}{7} = \frac{3}{8}.$$

(And indeed, it makes sense that the probability the second ball is red is the same as the probability the first one is red, provided we don't know anything about what color the first ball is.) However, these events would be independent if we replaced the first ball before drawing the second ball, in the sense that then the probability the second ball is red would not depend upon whether the first ball is red or not.

To give the formal definition of this sense of independence requires defining *conditional probabilities*. This is not hard, but we will not do it as there is is an alternative, equivalent way to state the notion of independence of two events, and this alternative formulation will actually be more useful for us. Namely, if two events are independent, then the probability of both of them happening is equal to the product of the probabilities of each of them happening separately. For instance, in our urn example, if we replace the first ball we draw before drawing the second one, then the probability that both balls are green is just the probability that the first is green times the probability that the second is green, i.e., $\frac{3}{8} \cdot 38$. Here is the formal definition.

Definition 2.2.7. *Let $A, B \subset S$ be two events in a probability space (S, P). Then A and B are* **independent** *if $P(A \cap B) = P(A)P(B)$.*

Note $A \cap B$ represents the event that both A and B happen. (Similarly, $A \cup B$ represents the even that either A or B, or both, happen.)

Example 2.2.8. *Consider our fair die roll example: $S = \{1, 2, 3, 4, 5, 6\}$ and $P(s) = 1/6$ for all $s \in S$. Let $A = \{2, 4, 6\}$ be the event of an even die roll, and $B = \{1, 2\}$ be the event of rolling ≤ 2. Then*

$$P(A \cap B) = P(2) = \frac{1}{6} = \frac{1}{2} \cdot \frac{1}{3} = P(A)P(B).$$

Hence A and B are independent events.

We remark that it is the property of independence that makes most random number generators on computers fail to be truly random—when we call our random number generator many times, the results should look pretty close to the uniform distribution, but the results are not actually independent. For instance, the simplest kind of random number generators repeat, although maybe not until you run them 4 billion times. This is an issue for some applications (e.g., cryptography) where (close to) true randomness is important, but it does not really for simple modeling/simulation like what we will do.

2.2.2 Erdös–Rényi Model

Notions of random graphs were introduced by Paul Erdös and Alfréd Rényi, and independently by Edgar Gilbert, in 1959. Two models were proposed, and these both now go by the name of the **Erdös–Rényi model.**

Here is the first model.

Definition 2.2.9. *Let $n \geq 1$ and $0 \leq M \leq n(n-1)/2$. The $G(n, M)$* **model for random graphs** *is the probability space (S, P), where S denotes the set of simple undirected graphs G on $\{1, 2, \ldots, n\}$ with M edges (or equivalently, the set of all subgraphs G of K_n with n nodes and M edges) and each graph $G \in S$ is assigned equal probability $P(G) = 1/|S|$.*

Let's explain how to compute $P(G)$ in terms of n and M.

Recall the following elementary fact from discrete mathematics: the *binomial coefficient* $\binom{n}{m} = \frac{n!}{m!(n-m)!}$, often read "$m$ choose n", is the number of ways to choose m distinct objects out of n total (distinct) objects. For instance, the maximum number of possible edges in a (simple undirected) graph on n nodes is $\binom{n}{2} = \frac{n(n-1)}{2}$, which is the number of way to choose 2 out of n vertices. Similarly, the maximum number of possible triangles in a graph is $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$. Note $\binom{n}{m} = \binom{n}{n-m}$—choosing m objects to include in something is the same as choosing $n - m$ objects to exclude from something.

Thus the number of graphs in S in the $G(n, M)$ model is simply the number of edges in K_n choose M, i.e., $\binom{n(n-1)/2}{M}$, as we just need to choose which M edges to include in our graph. Hence

$$P(G) = \frac{1}{\binom{n(n-1)/2}{M}}, \quad G \in S.$$

It is not hard to implement an algorithm to generate $G(n, M)$ graphs in Python (they are already implemented in Sage). Here is pseudocode

Pseudocode

```
set V = { 1, 2, ..., n }
set Eall = []
for i from 1 to n:
    for j from i+1 n:
         append {i, j} to Eall
set E = M randomly chosen edges from Eall
return G = (V, E)
```

That is, first we generate all possible undirected edges `Eall`, and then we randomly choose M of them to include in our graph. This is the only tricky part, but there is a command `shuffle` in the Python `random` module which makes this easy. For example, here is an example of how we can randomly choose 5 elements from a list.

Python 2.7

```
>>> from random import shuffle
>>> l = range(20)
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> shuffle(l)
>>> l
```

```
[6, 4, 9, 3, 18, 13, 12, 16, 19, 11, 17, 10, 0, 15, 7, 1, 2, 8, 5, 14]
>>> l[:5]
[6, 4, 9, 3, 18]
```

Now we'll present the second Erdös–Rényi random graph model.

Consider the following random procedure for making a graph $G = (V, E)$. Fix a probability $0 \leq p \leq 1$.

Pseudocode

```
set V = { 1, 2, ..., n }
set E = { }
for i from 1 to n:
    for j from i+1 to n:
        with probability p, add {i, j} to E
return G = (V, E)
```

In other words, we start with n vertices, and for each (unordered) pair of vertices $\{i, j\}$, we include the edge $\{i, j\}$ in our graph with probability p. (Recall, we saw how to do something with probability p in Python/Sage in Section 2.2.1.) Note in the pseudocode above we take $i + 1 \leq j \leq n$ to ensure that we loop through each unordered pair of distinct vertices $\{i, j\}$ exactly once.

This is the random process for the $G(n, p)$ model. Note the $G(n, p)$ model models a random generation of graphs on n nodes where the probability of link formation is p, independent of the choice of the link. To properly analyze the $G(n, p)$ model, we need to be able to calculate the probability of getting a given graph G of order n.

Let G be a random $G(n, p)$ graph. For $1 \leq i < j \leq n$, let $A_{i,j}$ denote the event that the edge $\{i, j\}$ is included G, and $B_{i,j}$ the event that the edge $\{i, j\}$ is not included in G. Say G has m edges: $\{i_1, j_1\}, \ldots, \{i_m, j_m\}$. Let $\{i_{m+1}, j_{m+1}\}, \ldots, \{i_N, j_N\}$ denote the remaining pairs of non-edges, where $N = n(n-1)/2$. Since the $A_{i,j}$'s and $B_{i',j'}$'s are all pairwise independent events, we see

$$P(A_{i_1,j_1} \cap A_{i_2,j_2} \cap \cdots \cap A_{i_m,j_m}) \cap B_{i_{m+1},j_{m+1}} \cap \cdots \cap B_{i_N,j_N}) =$$
$$P(A_{i_1,j_1})P(A_{i_2,j_2}) \cdots P(A_{i_m,j_m})P(B_{i_{m+1},j_{m+1}}) \cdots P(B_{i_N,j_N}) = p^m(1-p)^{N-m}.$$

Because this list of edges and non-edges determines G, we get that $P(G) = p^m(1-p)^{N-m}$, i.e., the probability only depends on how many edges m has.

With this in mind, we can also define of the $G(n, p)$ model in terms of the probability space.

Definition 2.2.10. *Let $n \geq 1$ and $0 \leq p \leq 1$. The $G(n, p)$* **model for random graphs** *is the probability space (S, P) where S is the set of all (simple undirected) graphs on $\{1, 2, \ldots, n\}$, where for $G \in S$ with m edges, we define the probability function to be $P(G) = p^m(1-p)^{n(n-1)/2-m}$.*

Note, if $p = 0.5$, then $P(G) = p^m(1-p)^{n(n-1/2)-m} = (0.5)^m(0.5)^{n(n-1)/2-m} = 0.5^{n(n-1)/2}$, independent of m.

Example 2.2.11. *Let $n = 3$ and $p = 0.5$. On $V = \{1, 2, 3\}$, there is $\binom{3}{0} = 1$ graph with 0 edges, $\binom{3}{1} = 3$ graphs with 1 edge, $\binom{3}{2} = 3$ graphs with 2 edges and $\binom{3}{3} = 1$ graph with 3 edges. Let*

A_m denote the subset of graphs on V with m edges. Hence, in the $G(n,p) = G(3, 0.5)$ model.

$$P(A_0) = P(0 \text{ edges}) = 1 \cdot (0.5)^3 = 0.125$$
$$P(A_1) = P(1 \text{ edge}) = 3 \cdot (0.5)^3 = 0.375$$
$$P(A_2) = P(2 \text{ edges}) = 3 \cdot (0.5)^3 = 0.375$$
$$P(A_3) = P(3 \text{ edges}) = 1 \cdot (0.5)^3 = 0.125.$$

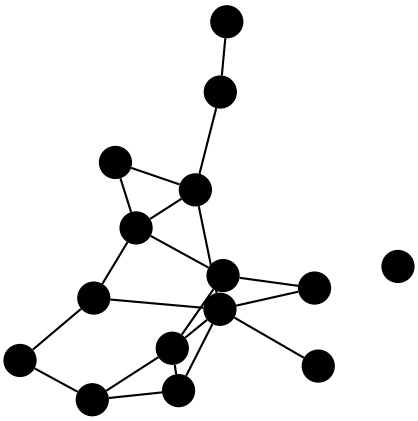
The $G(n, M)$ and $G(n, p)$ models are clearly different (the $G(n, p)$ model can result in any possible number of edges), but in many ways they behave rather similarly, particularly for large n. These are both models where edge formations are independent—we can view the $G(n, M)$ models as randomly distributing M edges to the $n(n-1)/2$ unordered pairs of distinct vertices, with each pair getting equal consideration. Also note that in either model, all graphs with a fixed number of edges m are equiprobable—it's just that in the $G(n, M)$ model this probability is 0 unless $m = M$. However, the $G(n, p)$ model is more commonly studied. Indeed, if people talk about random graphs without any further qualification, they probably have in mind the $G(n, p)$ model.

Let's look at our two favorite social networks, and see how well these can be modeled by random graphs. Let's first try this with the $G(n, M)$ model, since there's less work for us to do here. Denote the Florentine families network by F, and the karate club graph by K.

Start with F. Since F has 15 nodes and 20 edges, we can generate a random $G(n, M) = G(15, 20)$ graph in Sage as follows

Sage 6.1
```
sage: G = graphs.RandomGNM(15,20)
sage: G.show()
```

Then maybe you'll get something that looks like this.



This particular example isn't connected, but almost is (only 1 isolated node), and often this $G(n, M)$ random graph will be (we'll be more precise below). If we compare some landmarks of this graph G with F, they seem reasonably similar:

Sage 6.1
```
sage: F.degree_histogram()
[0, 4, 2, 6, 2, 0, 1]
sage: G.degree_histogram()
[1, 2, 4, 4, 3, 0, 1]
sage: F.cliques_maximum()
```

```
[[0, 7, 8], [0, 8, 11], [1, 2, 13]]
sage: G.cliques_maximum()
[[1, 2, 14], [1, 5, 14], [6, 8, 11]]
sage: F.cluster_transitivity()
0.19148936170212766
sage: G.cluster_transitivity()
0.1836734693877551
```

They both have 3 cliques of size 3, very close overall clustering, and fairly close degree distributions.

To be a bit more scientific, I wrote a function `testGNM(n,M)` which test some statistics about graphs in the $G(n, M)$ model. Namely, it generates 1000 random $G(n, M)$ graphs and counts the number of times they are connected and the averages of the following quantities: maximum degree, average distance (among connected graphs), average diameter (among connected graphs), max closeness centrality, max betweenness centrality, clique number and transitivity. Here is the result.

Sage 6.1

```
sage: testGNM(15,20)
Connected:   476 / 1000 times
Average max degree:   5.249
Average distance:   2.5818927571
Average diameter:   5.4243697479
Closeness centrality:    0.530094239059
Betweenness centrality:    0.353480946831
Average clique number:   2.985
Average transitivity:   0.173241028715
```

To compare, F has maximum degree 6, average distance 2.4857..., diameter 5, clique number 3, max closeness centrality 0.56, max betweenness centrality 0.52, and transitivity 0.1914... All in all, not so far off, except perhaps for betweenness centrality, so the $G(n, M)$ models many features of F rather accurately.

What would it mean if the $G(n, M)$ model is a good model for how the network F formed? It might me that rather than the de Medici family having blessed foresight in forging connections to let them rise to the top, maybe they just happened to be the lucky node that was most central. If one generates a few more examples of random graphs, then we'll see the above degree distribution sometimes looks like our previous example, but not most of the time. For another randomly chosen G, one gets the degree histogram `[0, 2, 4, 6, 3]`, where there are many vertices with relatively high degree centrality. Further, from doing a few trials, and looking at betweenness centrality, it seems quite unusual for one node to have betweenness centrality much higher than all other nodes, as the de Medicis do in F. This supports the hypothesis that indeed the de Medicis (as well as other families) did not choose their connections at random, but selected them strategically (and wisely in the case of the de Medicis).

Now let's consider K, which has 34 nodes and 78 edges. Again, we can run our program to test some statistics on a $G(n, M) = G(34, 78)$ model.

Sage 6.1

```
sage: testGNM(34,78)
Connected:   796 / 1000 times
Average max degree:   8.979
Average distance:   2.41009190337
Average diameter:   4.78140703518
```

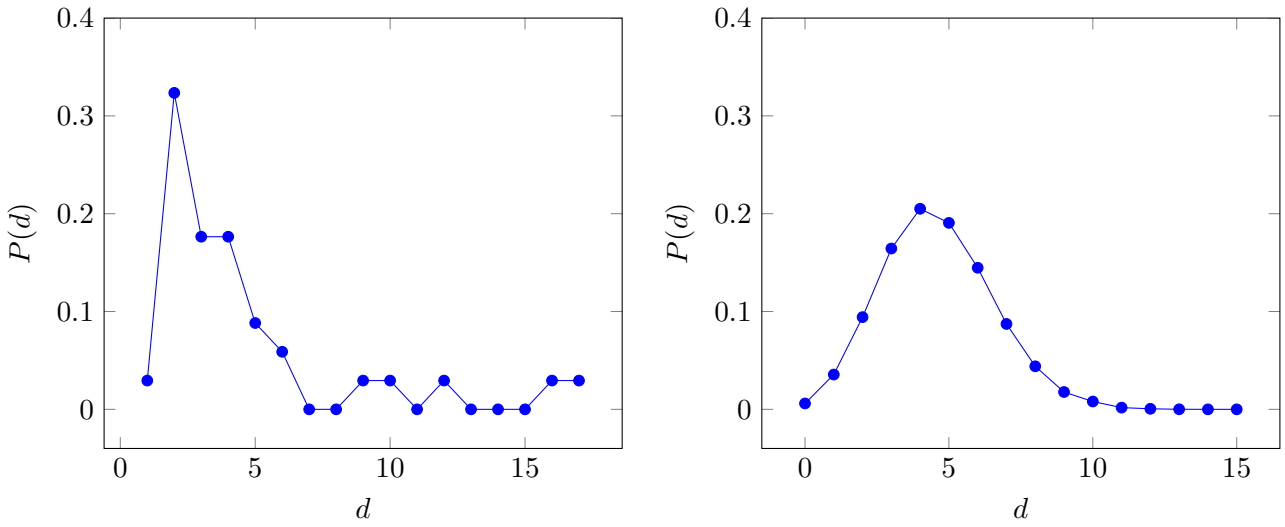


Figure 2.6: Degree distibution for K and average degree distribution for 1000 $G(34, 78)$ graphs

```
Closeness centrality:    0.520352742539
Betweenness centrality:    0.152535379873
Average clique number:   3.24
Average transitivity:   0.136935568331
```

Compare this to K, which has max degree 17, average distance 2.408..., diameter 5, max closeness centrality 0.5689..., max betweenness centrality 0.4376..., clique number 5 and transitivity 0.2556... Some of the statistics are close, but a few are far off: max degree, betweenness centrality, clique number and transitivity.

In particular, the degree distribution in the $G(n, M)$ model is way off from K. To see this clearly, I computed the average degree distribution for 1000 $G(34, 78)$ graphs and plotted this next to the degree distribution for K in Figure 2.6. If $G(n, M)$ is a good model for K (that is, if K looks similar to an "average" graph from $G(n, M)$ rather than a low probability one), these graphs should have pretty close to the same shape, but the $G(n, M)$ degree distribution has a much gentler, rounder slope, with peak around degree 4, rather than degree 2.

Unlike the Florentine families graph, where at least sometimes the degree distributions of our random graphs matched pretty closely, they never match closely for the Karate club graph. In fact, the max degee ever achieved in this trial of the $G(34, 78)$ model was 13, which only occured twice (the max for K is 17). Part of this may be attributed to the size of the graphs—the Florentine families graph is much smaller, so there is less possible variation for networks of this size. However, part of it is undoubtedly due to the difference in the nature of the links in F and K—the links in F are marriages, likely largely chosen strategically, whereas the links in K are friendships, where one expect things like clustering and other small world phenomena.

Thus we can safely conclude that the friendships formed in K cannot be explained (at least not entirely) by random, independent formation (independent chance encounters). We can't explain the large hubs (one degree 16 node and one degree 17 node) by random formation—they are distinguished individuals in the clubs, the instructor and student founder. Moreover, the amount of sizable cliques and clustering/transitivity in K is not present in a random $G(n, M)$ graph.

What about the $G(n,p)$ model? We will get very similar results as with the $G(n,M)$ model, but let us explain how we can compare a given graph with the $G(n,p)$ model. The difference here is that while it was easy to see what M should be in the $G(n,M)$ model, it is not immediately obvious how to chose the appropriate probability p for edge formation.

Proposition 2.2.12. *The average, or expected, number of edges in a graph in the $G(n,p)$ model is*

$$\sum_{m=0}^{n(n-1)/2} m \binom{n(n-1)/2}{m} p^m (1-p)^{n(n-1/2)-m}.$$

Proof. To average a random quantity over a discrete probability space, instead of just summing up all values and dividing by the number of entries, we sum up all values weighted by the probability of obtaining that value. So the expected number of edges in a graph in the $G(n,p)$ model is

$$\sum_{m=0}^{n(n-1)/2} P(m \text{ edges}) \cdot m.$$

Now the probability of m edges is simply $p^m(1-p)^{n(n-1/2)-m}$ times the number of graphs with m edges, which we saw above is $\binom{n(n-1)/2}{m}$. □

Using this proposition, we can with trial and error, find a value of p that gives us the desired expected number of edges. For example, when $n = 15$, $p = 0.19$ gives an expected 19.95 edges, so we can try to model F with $G(n,p) = G(15, 0.19)$. Similarly, with $n = 34$, $p = 0.139$ gives an expected 77.979 edges, so this is reasonable choice to try to model K. I wrote a Sage function `testGNP` which is the analogue of `testGNM` for the $G(n,p)$ model, and you can see the results below are very similar to those for $G(n,M)$.

Sage 6.1

```
sage: testGNP(15,0.19)
Connected:   426 / 1000 times
Average max degree:   5.252
Average distance:   2.45504135927
Average diameter:   5.10328638498
Closeness centrality:    0.516331967533
Betweenness centrality:    0.340416128774
Average clique number:   2.944
Average transitivity:   0.171173755959

sage: testGNP(34,0.139)
Connected:   793 / 1000 times
Average max degree:   9.043
Average distance:   2.40368150011
Average diameter:   4.75283732661
Closeness centrality:    0.520815091198
Betweenness centrality:    0.154727157339
Average clique number:   3.255
Average transitivity:   0.136305968809
```

2.2.3 Preferential Attachment Models

We saw in the last section that the Erdös–Rényi model is not suitable for modeling some friendship networks, such as the karate club graph. There are a couple of issues here. One, there is not enough clustering in this model of random graphs. Two, we don't see nodes of high degree.

Let me briefly mention a couple of other simple models to deal with these specific issues.

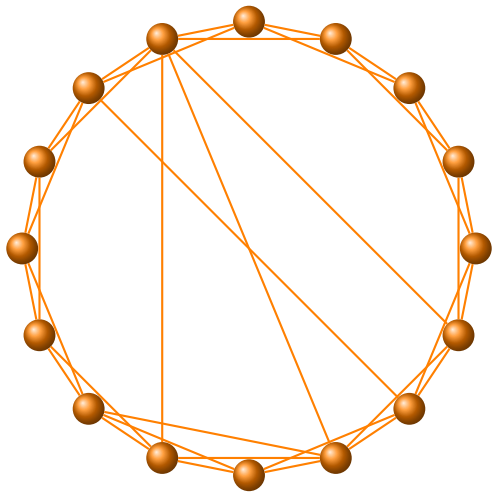
The **Watts–Strogatz model** is one model to generate random graphs with a specified average degree and large clustering. Here we fix a number of nodes n, the average degree κ, and a probability $0 \leq \beta \leq 1$. The algorithm is as follows Arrange the n nodes in a circle, and initially connect each node u to the κ nodes closest to u on the circle (say $\lfloor \kappa/2 \rfloor$ nodes on the left and $\lceil \kappa/2 \rceil$ nodes on the right). So for $\kappa = 2$, this is just a cycle graph at this stage. (In general, it is a kind of generalization of cycle graphs called *circulant graphs*.) Now, for each edge (u, v) in the graph, with probability β replace it with an edge (u, v') where v' is chosen at random among all other vertices.

For whatever reason, Sage has not implemented the Watts–Strogatz model, but the closely related Newman–Watts–Strogatz model, which is just like the Watts–Strogatz model except that instead of randomly replacing existing edges, it simply adds new ones. The can be accessed via the function `graphs.RandomNewmanWattsStrogatz(n,kappa,beta)` in Sage as follows.

Sage 6.1

```
sage: n = 16
sage: G = graphs.RandomNewmanWattsStrogatz(n,4,0.1)
sage: pos_dict = {}
sage: for i in range(n):
....:     x = float(cos(pi/2 + ((2*pi)/n)*i))
....:     y = float(sin(pi/2 + ((2*pi)/n)*i))
....:     pos_dict[i] = [x,y]
....:
sage: p = G.graphplot(pos = pos_dict)
sage: p.show()
```

While we could've just done `G.show()` after the second line, Sage does not naturally arrange these vertices in a circle, so we did a little more work to plot it like this. This code should give a graph that looks something like this.



Note that in the Watts–Strogatz model, we will get precisely $n\kappa$ edges. If $\beta = 0$, we get a κ-regular graph with n cliques of order $\lfloor \frac{\kappa}{2} \rfloor$. On the other hand, as $\beta \to 1$, the Watts–Strogatz random

graphs get closer to the random graphs in a $G(n, M) = G(n, n\kappa)$ model. Hence this model gives us a hybrid of regular graphs with a lot of clustering and Erdös–Rényi random graphs. Consequently, we will almost never get nodes of high degree like in the karate club graph. Another issue is that the Watts–Strogatz model will almost always have a cycle of length n, unless β is quite high, and this is not typically expected in social networks (e.g., look at the Florentine families or karate club graphs).

A completely different idea for generating random graphs is **preferential attachment**, that is, nodes tend to form links with nodes that have already high degree. The idea comes from consideration of citation networks. Here we consider graph the nodes represent certain publications—e.g., all mathematics publications as indexed by the American Mathematical Society (MathSciNet)—and we draw a directed edge from paper A to paper B if paper A cites paper B. This is similar in spirit to webpage hyperlink graphs.

Note that both citation networks and hyperlink graphs are dynamic in nature—new publications and new webpages are always being born. Right away, this is completely different from the Erdös–Rényi and Watts–Strogatz models, which are by nature static, in that the number of nodes do not change in the generation process. (However, we could generate graphs in the $G(n, p)$ model by adding nodes at each stage.)

Preferential attachment is based on the following idea. When I do research, how do I find what is already known about a subject? I look at other research papers or books, or go to conferences, or talk to experts. If another paper in this area is highly cited, I am likely to come across it and/or find it relevant, and likely to cite this also. Whereas, if a paper is not cited much, I am less likely to both come across this paper and find it relevant. If we're a bit more precise about this, we'll see this heuristic predicts that the probability I cite another paper is proportional to the number of citations that paper already has, i.e., that paper's in degree in the citation network. The exact same heuristic makes sense for hyperlink graphs. Furthermore, it is not hard to see this heuristic predicts a scale-free (power law) degree distribution.

One well-known preferential attachment model is the **Barabási–Albert model**. The algorithm begins with an initial connected network of m_0 nodes. There are fixed parameters of n (total number of nodes, or papers), and m (the "out degree" or number of citations each new paper will make). At each stage (until there are n nodes), we add a new node to the network and connect it to m existing nodes chosen at random with respect to a "preferential" probability distribution. Here we can use either directed or undirected edges—directed edges makes sense for a citation/hyperlink network, but the Barabási–Albert model just works with undirected edges.

The actual description of the algorithm for this preferential attachment by Barabási–Albert in their papers is a little vague but here is how I interpret it. Suppose we are at a stage with n_0 nodes numbered $1, 2, \ldots, n_0$. Let d_i be the degree of node i. Set

$$p_i = \frac{d_i}{\sum d_i}.$$

Add a new vertex $n_0 + 1$, and do the following m times. Connect $n_0 + 1$ to one of the vertices in $\{1, 2, \ldots, n_0\}$, where vertex i gets selected with probability p_i. This choice can be practically implemented as follows. Choose a random number x in $[0, 1]$. Divide $[0, 1]$ into n_0 intervals, $I_1, \ldots, I_{n_0}$ where interval I has length p_i. For instance, if $n_0 = 3$, we may take the following partition:

$$[0, 1] = [0, p_1) \cup [p_1, p_1 + p_2) \cup [p_1 + p_2, p_1 + p_2 + p_3 = 1] = I_1 \cup I_2 \cup I_3.$$

If the random number x lies in I_i, connect $n_0 + 1$ to p_i.

The ambiguity here, if $m > 1$, is what to do if you're trying to add an edge you've just added, e.g., if you randomly pick the edge $(n_0 + 1, 1)$ twice. One can either allow multi-edges or pick another edge. Allowing multi-edges makes the analysis easier in some ways, but it seems that at least the Sage implementation chooses m distinct edges. (I'm not sure what Barab'asi and Albert originally did.) Picking m distinct edges can be done in one of two ways—(1) if we happen to choose an edge we've just added, we can just randomly choose again until we get a new edge; or (2) before a new edge is selected, we can update the probabilities for connecting to each vertex—that is, compute the probabilities as above, but simply leave out all the vertices we already connected to $n_0 + 1$. For instance, in approach (2), say $n_0 = 3$, $m = 2$ and we just added an edge from the new vertex 4 to vertex 1. Then we set

$$p_1 = 0, \quad p_2 = \frac{d_2}{d_2 + d_3}, \quad p_3 = \frac{d_3}{d_2 + d_3}$$

and now add an edge from vertex 4 to vertex i with probability p_i.

It is convenient to be able to do this without starting with an initial connected network of m_0 nodes, but just start with the parameters n and m. Then one can start with a initial empty graph on m nodes, and connect the next node added to each of these m nodes. Then proceed as above. This appears to be how the algorithm is implemented in Sage.

While the algorithm is given for a fixed n, this model is dynamic as the networks are generated by adding more nodes. Furthermore this graph generation can be done in stages, and the output of each stage can be put in as the "input graph" in the next stage without affecting the probability distribution in the model. For instance if we generate a preferential attachment graph G_{99}, with $(n, m) = (99, 3)$, we can generate a graph G_{100} with $(n', m) = (100, 3)$ in this model by starting by using G_{99} as our initial network of m_0 nodes, and adding 1 node in the above process. Note we cannot do this sort of thing in the Watts–Strogatz model or the $G(n, M)$ model, though we can for the $G(n, p)$ model.

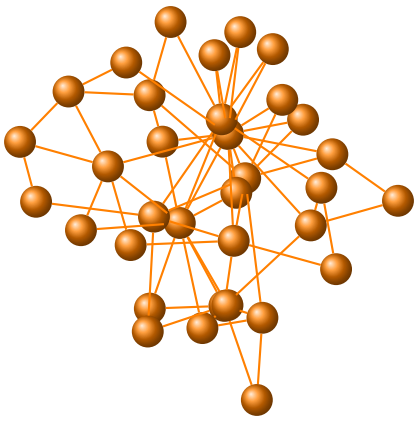
Assuming this implementation, the Barabási–Albert model always has $(n - m)m$ edges. In addition, it will always be connected, hence it is a tree for $m = 1$.

Now if we want to try to model the karate club graph this way, we can take $n = 34$ and either $m = 2$ or $m = 3$. These will give us 64 and 93 edges respectively. Remember the karate club graph as 78 edges.

Here is an attempt with $m = 2$.

Sage 6.1

```
sage: G = graphs.RandomBarabasiAlbert(34,2)
sage: G.degree_histogram()
[0, 0, 17, 5, 4, 1, 2, 2, 1, 0, 1, 0, 0, 0, 1]
sage: len(G.degree_histogram()) - 1
14
sage: G.show()
```



We see this example has maximum degree 14, which is higher (by 1) than what we ever got with 1000 random $G(n, M) = G(34, 78)$ graphs. However, one can get higher maximum degrees in this model. Here are the maximum degrees and degree distributions for 10 such random graphs.

Sage 6.1

```
sage: for i in range(10):
....:     dh = graphs.RandomBarabasiAlbert(34,2).degree_histogram()
....:     print len(dh) - 1, ":  ", dh
....:
12 :   [0, 0, 12, 9, 6, 2, 1, 0, 2, 1, 0, 0, 1]
10 :   [0, 1, 14, 8, 3, 2, 0, 0, 2, 3, 1]
11 :   [0, 0, 14, 7, 4, 1, 4, 2, 0, 1, 0, 1]
13 :   [0, 0, 18, 6, 2, 2, 1, 1, 0, 1, 1, 1, 0, 1]
17 :   [0, 0, 13, 10, 4, 2, 1, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]
13 :   [0, 0, 16, 6, 4, 1, 4, 1, 0, 0, 0, 0, 0, 2]
17 :   [0, 0, 18, 5, 5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1]
15 :   [0, 0, 13, 12, 1, 2, 0, 3, 2, 0, 0, 0, 0, 0, 0, 1]
14 :   [0, 0, 16, 4, 5, 3, 3, 0, 1, 1, 0, 0, 0, 0, 1]
10 :   [0, 0, 17, 4, 3, 1, 5, 0, 2, 1, 1]
```

We see the maximum degree varies a lot, and we happened to get a couple instances where the maximum degree is the same as that of the karate club graph, however it seems unlikely that we get one node of degree 16 and one of degree 17. Of course these graphs have fewer edges than our karate club graphs. If instead we look at degree distributions for preferential attachment graphs with $m = 3$, we get something like the following.

Sage 6.1

```
sage: for i in range(10):
    dh = graphs.RandomBarabasiAlbert(34,3).degree_histogram()
    print len(dh) - 1, ":  ", dh
....:
18 :   [0, 0, 0, 9, 9, 7, 3, 0, 2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1]
16 :   [0, 0, 1, 13, 6, 3, 2, 1, 3, 1, 0, 1, 0, 1, 1, 0, 1]
16 :   [0, 0, 0, 11, 5, 8, 2, 1, 2, 2, 0, 1, 0, 1, 0, 0, 1]
21 :   [0, 0, 1, 15, 5, 2, 3, 1, 0, 2, 1, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1]
18 :   [0, 0, 0, 14, 5, 5, 1, 3, 2, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1]
16 :   [0, 0, 1, 11, 6, 3, 2, 4, 4, 0, 0, 1, 0, 1, 0, 0, 1]
17 :   [0, 0, 0, 10, 11, 2, 3, 1, 0, 1, 4, 1, 0, 0, 0, 0, 0, 1]
12 :   [0, 0, 0, 13, 4, 5, 3, 1, 1, 3, 0, 2, 2]
```

```
17 :   [0, 0, 1, 15, 3, 4, 2, 2, 1, 2, 0, 1, 1, 0, 0, 1, 0, 1]
16 :   [0, 0, 1, 13, 5, 4, 4, 1, 1, 0, 1, 0, 1, 0, 2, 0, 1]
```

Here we see instances where we get a couple of nodes of large degree, though now have too few nodes of degree 2 (the karate club had 11 such nodes, and 1 of degree 1).

This suggests that the Barabási–Albert model is not too bad at modeling the degree distribution for the karate club graph, though ideally we would want to choose something like $m = 2.5$ in this model. While m has to be an integer, we could modify this algorithm to allow fractional m. In this case, $m = 2.5$ would mean that at half the stages in our algorithm we add 2 edges to new vertices, and the other half of the time we add 3 edges.

In fact, analysis of this algorithm shows that the degree distribution follows a power law given roughly by

$$P(d) = \begin{cases} \frac{2m^2}{d^3} & d \geq m \\ 0 & \text{else.} \end{cases}$$

This is not exactly true of course for fixed n—for instance $P(d) = 0$ for $d > n$ and some of the initial m_0 vertices may have degree less than m—but this is the limit of the degree distribution as $n \to \infty$.

While we may view the Barabási–Albert model as a relatively good fit for the degree distribution of the karate club graph, there are other ways in which it is less than ideal. First, we cannot choose appropriate m to get the desired number of edges, but this can be remedied by allowing fractional values of m. Second, whenever $m > 2$ there will never be many nodes of degree $< m$, but nodes of small degree are common in many social networks. Third, graphs in this model tend to have relatively low clustering compared with friendship networks.

Exercises

Exercise 2.2.1. *Consider the process of flipping a fair coin twice. Write down a probability space S to model this process, and say what the probability function P is. For the following pairs of events A and B, write the events A and B explicitly as subsets of S and determine (with proof), whether A and B are independent or not:*

(i) A is getting heads on the first flip, and B is getting heads on the second flip

(ii) A is getting heads on the first flip, and B is getting heads on both flips

(iii) A is getting the same result on both flip, and B is getting different results on each flip

(iv) A is getting heads on the first flip, and B is getting heads on at least one flip

Exercise 2.2.2. *Write a function* `GnM(n,M)` *in Python that returns the adjacency matrix for a random graph in the $G(n, M)$ model.*

Exercise 2.2.3. *Write a function* `Gnp(n,p)` *in Python that returns the adjacency matrix for a random graph in the $G(n, p)$ model.*

Exercise 2.2.4. *Let $n = 4$ and $p = 0.5$. For each $0 \leq m \leq n(n-1)/2$, compute the probability a graph in the $G(n, p)$ model has m edges.*

Exercise 2.2.5. *Write a function* `Prmedge(n,p,m)` *in Sage that returns the probability a graph in the $G(n, p)$ model has m edges. Then for $n = 5$, each $p \in \{0.1, 0.2, 0.3\}$, and each $0 \leq m \leq 10$, compute the probability that a graph in the $G(n, p)$ model has m edges.*

Exercise 2.2.6. *Write a function* `testGnpconn(n,p)` *in Sage that generates 1000 graphs in the* $G(n,p)$ *model and outputs (prints, returns, or both, your choice) the number that are connected. For* $n = 20$*, how large should* p *be (to 2 decimal places) in order for random* $G(n,p)$ *graphs to be connected at least 99% of the time.*

Exercise 2.2.7. *Write a Sage function* `WattsStrogatz(n,kappa,beta)` *that generates and returns a graphs in the Watts–Strogatz model.*

Exercise 2.2.8. *Write a Sage function* `PrefAttach(n,m)` *that generates a Barabási–Albert type graph, but allows* m *to be fractional. Precisely if we write* $m = m' + p$ *where* m' *is an integer and* $0 \leq p < 1$*, at each stage we add* m' *edges to our new node with probability* $1 - p$*, and* $m' + 1$ *nodes with probability* p*.*

Generate 20 such graphs with $n = 34$ *and* $m = 2.5$*, and print out the maximum degrees and degree distributions. Does this seem to match closely with the degree distribution of the karate club graph?*

Exercise 2.2.9. *(i) For Barabási–Albert graphs with* $(n,m) = (34,2)$*, generate 100 random graphs and compute the average of the following quantites: diameter, maximum closeness centrality, maximum betweenness centrality, and transitivity. Compare these with the corresponding values for the karate club graph.*

(ii) Do the same for $(n,m) = (34,3)$*.*

Exercise 2.2.10. *(i) Design your own algorithm for generating random graphs to model friendship network (not necessarily the karate club graph). Explain your reasoning behind your algorithm.*

(ii) Code up your algorithm in Sage. By computing some examples, see how well it models some features of the karate club graph, and compare it with the other models we've studied.