hw02

Charles Valentine

November 4, 2016 6PM

Question #1

Note: Python 3 code can be found at the bottom of this document.

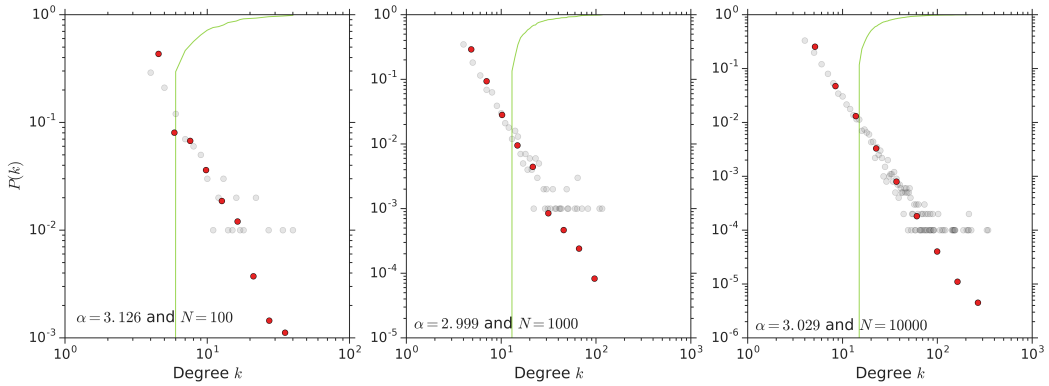

Figure 1: Degree distributions and powerlaw fits for the Barabasi-Albert network model with $m = 4$.

It is shown in Figure 1 that a powerlaw fit to the observations in the Barabasi-Albert network has an α value that converges to 3.0. This is congruent with the analytical derivations that show, due to growth and preferential attachment, the degree distribution can be modeled with a powerlaw distribution with $\alpha \approx 3.0$.

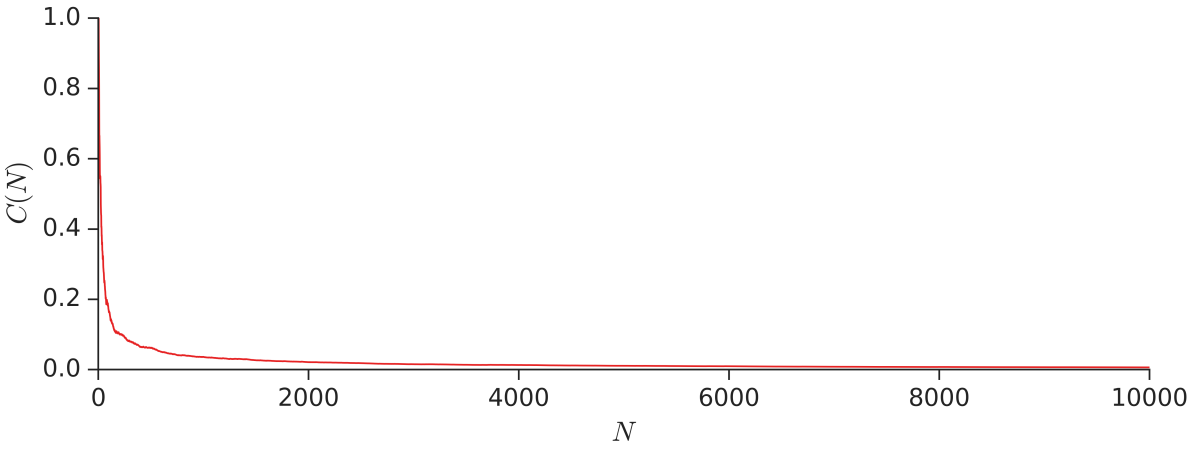

Figure 2: Change in the average clustering coefficient for the network through time ($t \approx N$).

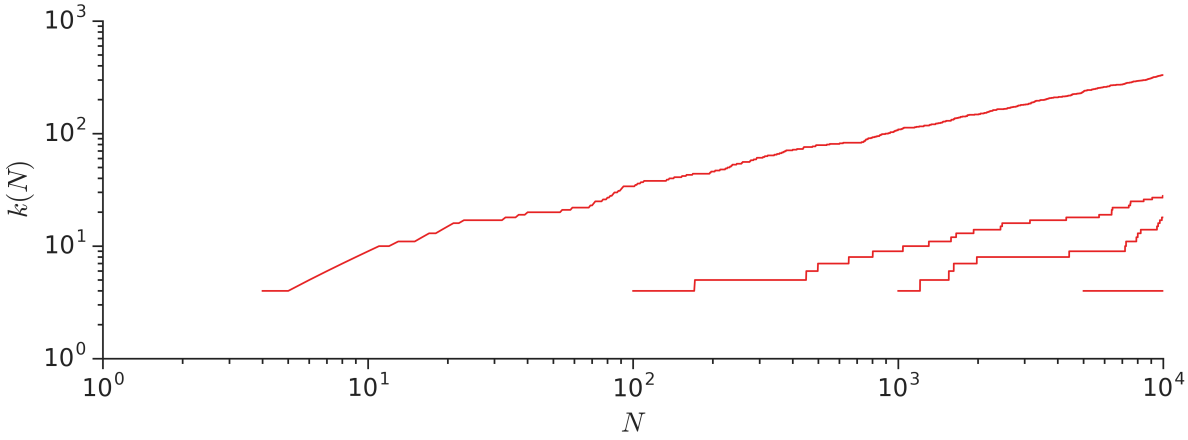

Figure 3: Local degree dynamics of four nodes.

Question #2

We have rendered this network with *Gephi* and calculated a few of the statistics of the network. First the average degree of the network is approximately 3.5. At first glance the degree distribution appears to look like a powerlaw might fit the sampled data, however, we do not have enough order of magnitudes preset, due to sampling size, to come to any robust conclusions. It is immediately noticeable that a few nodes are well connected. We postulate these nodes are principal investigators that work with many post-docs and other principal investigators in other laboratories. A few k-cliques exist which are likely research groups that always publish together.

The community structure has been inferred and colored for the giant component of the graph. It is tough to deduce if this structure fits any known metadata, without having any known metadata. One hypothesis is that communities may appear in sub-disciplines of network science. A second hypothesis which could be driving community structure is regional or university specific collaboration.

This network is highly fragmented with many isolated cliques. Therefore measures of centrality are inaccurate in determining the true behavior of this network because the distance between any two randomly chosen nodes is often infinity. The average clustering coefficient of this network (0.88) represents high clustering in the giant component and supports our hypothesis that their are communities in the underlying process of scientific collaboration.

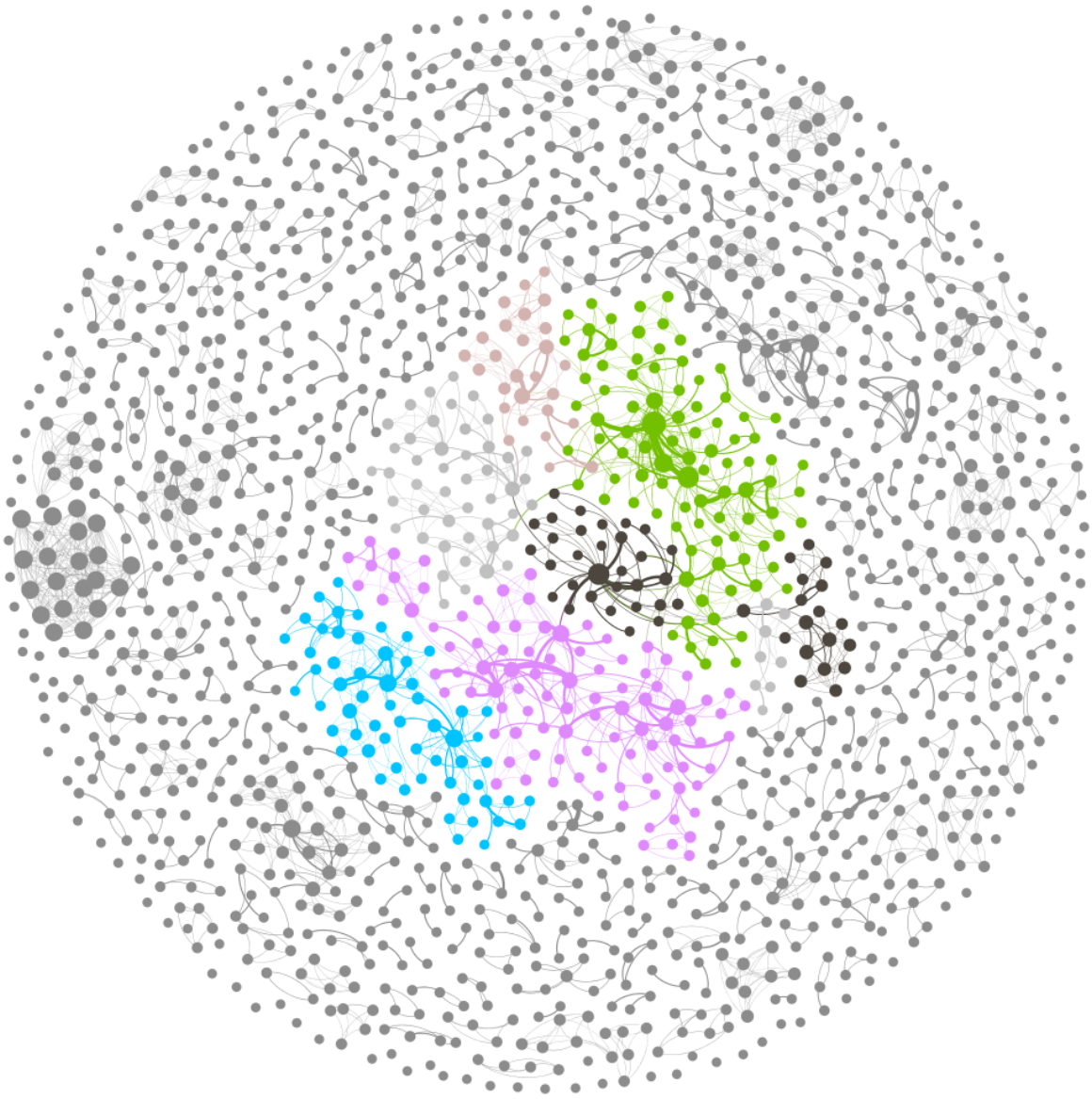

Figure 4: Network of coauthorship network of scientists working in network theory compiled by Mark Newman in 2006. Colors represent inferred communities in the giant component. The size of the nodes directly relates to their degrees.

Question #3

The probability of finding a node with degrees k and $k\prime$ at the end of a randomly selected link is $e_{kk\prime}$, the joint probability. We can determine the expected number of links between k and $k\prime$ *via* the following relationship:

$$E_{kk\prime} = e_{kk\prime}\langle k\rangle N \tag{1}$$

This simplifies to:

$$e_{kk\prime} = \frac{E_{kk\prime}}{\langle k\rangle N} \tag{2}$$

We can also represent the probability that, given a random link, there is a degree-k node at one determined end.

$$q_k = \frac{kp_k}{\langle k\rangle} \tag{3}$$

The probability of a degree k node can be represented as ${}^{N_k}/_{N}$. This simplifies the previous equation to:

$$q_k = \frac{k{}^{N_k}/_{N}}{\langle k\rangle} \tag{4}$$

The conditional probability $P(k\prime|k)$ that following a link of a k-degree node we reach a degree-$k\prime$ node can be represented as the total number of links from degree-k nodes to degree-$k\prime$ nodes ($E_{kk\prime}$) over the total number of links following k-degree nodes (kN_k).

$$P(k\prime|k) = \frac{E_{kk\prime}}{kN_k} \tag{5}$$

Using the above simplifications we can show that any network has the following property:

$$e_{kk\prime} = q_k P(k\prime|k) \tag{6}$$

$$\frac{E_{kk\prime}}{\langle k\rangle N} = \frac{k{}^{N_k}/_{N}}{\langle k\rangle} \cdot \frac{E_{kk\prime}}{kN_k} \tag{7}$$

$$\frac{E_{kk\prime}}{\langle k\rangle N} = \frac{kN_k\frac{1}{N}E_{kk\prime}}{\langle k\rangle kN_k} = \frac{E_{kk\prime}}{\langle k\rangle N} \tag{8}$$

$$\frac{E_{kk\prime}}{\langle k\rangle N} = \frac{E_{kk\prime}}{\langle k\rangle N} \tag{9}$$

Question #4

a. In the model presented to us in the assignment we can conclude that the number of links in a subnetwork L_C with m nodes is related to the degree k of that subnetwork k_C with the equation $L_C = \frac{k}{2}m$. The number of links in the entire network L is equal to the sum of all the links in the identical subnetworks and the additional pairwise links connecting all subnetworks as in the equation $\frac{k}{2}mC + C$. Finally, the total degree of each subnetwork is simpy the degree k multiplied by the nodes in the the subnetwork m in addition to the number of outoing links to other components as in the equation $km + (C - 1)$.

The modularity of a subnetwork can be calculated as follows:

$$M_C = \frac{L_C}{L} - (\frac{k_C}{2L})^2 \tag{10}$$

$$M_C = \frac{\frac{k}{2}m}{\frac{k}{2}mC + C} - \left(\frac{km + (C-1)}{2(\frac{k}{2}mC + C)}\right)^2 \tag{11}$$

The modularity of the entire partition is the sum of the modularity values for each subnetwork:

$$M = \sum_{C=1}^{n_C} \left[\frac{\frac{k}{2}m}{\frac{k}{2}mC + C} - \left(\frac{km + C - 1}{2(\frac{k}{2}mC + C)}\right)^2\right] \tag{12}$$

b. When merging pairs of subnetworks we must take into account differences in expressing the modularity of the partition. The links in the subnetwork becomes $km + 1$ and the total degree *per* subnetwork becomes $2(km + C - 1)$:

$$M = \sum_{C=1}^{n_C} \left[\frac{km + 1}{\frac{k}{2}mC + C} - \left(\frac{km + C - 1}{\frac{k}{2}mC + C}\right)^2\right] \tag{13}$$

c. I was unable to make the comparison using the equations above and therfore could note complete these followup questions!

Code for Question #1

```
%load_ext autoreload
%autoreload 2
%matplotlib inline
import powerlaw
import numpy as np
import networkx as nx
import matplotlib as mpl
from itertools import repeat
from collections import Counter
import matplotlib.pyplot as plt

mpl.rc('xtick', labelsize=14, color="#222222", direction='out')
mpl.rc('ytick', labelsize=14, color="#222222", direction='out')
mpl.rc('font', size=16)
mpl.rc('xtick.major', size=6, width=1)
mpl.rc('xtick.minor', size=3, width=1)
mpl.rc('ytick.major', size=6, width=1)
mpl.rc('ytick.minor', size=3, width=1)
mpl.rc('axes', linewidth=1, edgecolor="#222222", labelcolor="#222222")
mpl.rc('text', usetex=False, color="#222222")

def cleanup_chart_junk(ax):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.yaxis.set_ticks_position('left')
    ax.xaxis.set_ticks_position('bottom')
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
    return ax
```

```
to_plot = [1e2, 1e3, 1e4]
fig, axes = plt.subplots(1, len(to_plot), figsize=(len(to_plot) * 6, 6))

m = 4
source = m
size_end = 1e4
targets = list(range(m))
G = nx.complete_graph(n=m)

node_pool, C_t = [], []
node_k = {m: [], 100: [], 1000: [], 5000: []}

while source < size_end:
    G.add_edges_from([(source, target) for target in targets])
    node_pool.extend(list(targets))

    targets = set()
    while len(targets) < m:
        targets.add(np.random.choice(node_pool))

    node_pool.extend(list(repeat(source, m)))
```

```
    C_t.append(nx.average_clustering(G))
    source += 1

    for key in node_k.keys():
        if key < len(G):
            node_k[key].append(G.degree(key))

    if len(G) in to_plot:
        ax = axes[to_plot.index(len(G))]
        degrees = list(G.degree().values())

        x_data, y_data = zip(*Counter(degrees).items())
        y_data = [count / len(G) for count in y_data]

        # Get 10 logarithmically spaced bins between kmin and kmax
        bin_edges = np.logspace(np.log10(min(degrees)), np.log10(max(degrees)), num=10)

        # histogram the data into these bins
        x = 10**((np.log10(bin_edges)[1:] + np.log10(bin_edges)[:-1]) / 2)
        density, _ = np.histogram(list(G.degree().values()), bins=bin_edges, density=True)

        # Fit a powerlaw distribution to the samples we assume to have been drawn
        # from a powerlaw distribution
        fitted_pl = powerlaw.Fit(degrees)
        fitted_pl.plot_cdf(ax=ax, label="Powerlaw CDF", color='#97D54C')


        ax.plot(x_data,
                y_data,
                label="Data",
                alpha=0.2,
                color='grey',
                marker='o',
                linestyle='none', lw=0)
        ax.plot(x,
                density,
                label="Log-binned Data",
                color='#E62223',
                marker='o',
                linestyle='none',
                lw=0)
        ax.set_xlabel(r"Degree $k$")
        ax.set_yscale('log')
        ax.set_xscale('log')

        ax.text(ax.get_xlim()[0] * 1.2,
                ax.get_ylim()[0] * 1.4,
                ha='left',
                s=r'$\alpha = {}$ and $N = {}$'.format(round(fitted_pl.alpha, 3), len(G)))

axes[0].set_ylabel(r"$P(k)$")
axes[1].legend(loc=9,
               bbox_to_anchor=(0.5, -0.2),
```

```
                frameon=False,
                fontsize=15,
                numpoints=1,
                scatterpoints=1,
                ncol=3)
fig.suptitle("Degree Distribution and Powerlaw Fits\n\n", fontsize=28)
```

```
fig, ax = plt.subplots(1, 1, figsize=(10, 4))

ax.plot(range(m, size_end), C_t, color='#E62223')
ax.set_xlabel("$N$")
ax.set_ylabel('$C(N)$')
ax.set_title('Change in Clustering Coeff. through Time')
ax = cleanup_chart_junk(ax)

fig.tight_layout()
```

```
fig, ax = plt.subplots(1, 1, figsize=(10, 4))

for node, degrees in node_k.items():
    if len(degrees) == 0:
        continue
    ax.plot(range(node, size_end), degrees, color='#E62223')
    ax.set_xlabel("$N$")
    ax.set_ylabel('$k(N)$')
    ax.set_title('Degree Dynamics of {} Nodes'.format(len(node_k.values())))
    ax = cleanup_chart_junk(ax)
    ax.set_yscale('log')
    ax.set_xscale('log')

fig.tight_layout()
```