CS4423 - Networks

Prof. Götz Pfeiffer
School of Mathematics, Statistics and Applied Mathematics
NUI Galway

6. Power Laws and Scale-Free Graphs

Lecture 21: Power Laws

In [1]:
```
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
```

Degree Distribution

Recall **degree distribution**:

The **degree distribution** of an undirected graph $G = (X, E)$ is the function $k \mapsto p_k := n_k/n$, where $n = |X|$ and n_k is the number of nodes of degree k (and thus p_k is the probability that a random node $x \in X$ has degree k).

In an ensemble of graphs of order n, one sets $p_k := \overline{n_k}/n$, where $\overline{n_k}$ is the expected value of the random variable n_k over the ensemble of graphs.

In this sense, the degree distribution in a random $G(n, p)$ graph is **binomial** :

$$p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k},$$

or, in the limit $n \to \infty$ and $p \to 0$ with np constant, it is a **Poisson distribution**:

$$p_k = e^{-z} \frac{z^k}{k!},$$

where $z = np$.

A **power law** degree distribution is strikingly different:

$$p_k = ck^{-\gamma},$$

for certain constants c and γ. (Typically $2 \leq \gamma \leq 3$.)

In [2]:
```
def binomial(n, k):
    prd, top, bot = 1, n, 1
    for i in range(k):
        prd = (prd * top) // bot
        top, bot = top - 1, bot + 1
    return prd
```

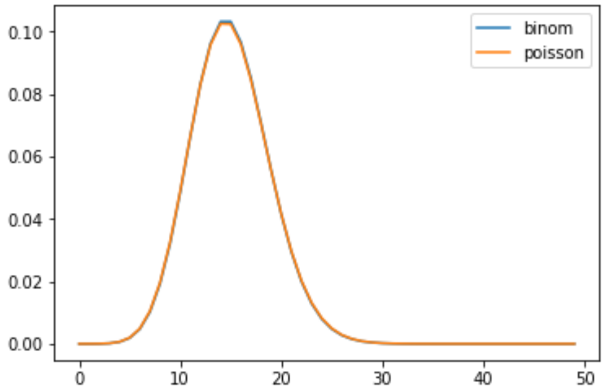
In [3]:
```
def b_dist(n, p, k):
    return binomial(n, k) * p**k * (1-p)**(n-k)
```

In [4]:
```
from math import exp, factorial
def p_dist(l, k):
    return exp(-l) * l**k / factorial(k)
```

In [5]:
```
n, p = 1000, 0.015
mm = 50
l = p * (n-1)
bb = [b_dist(n-1, p, k) for k in range(mm)]
pp = [p_dist(l, k) for k in range(mm)]
```

In [6]:
```
df = pd.DataFrame()
df['binom'] = bb
df['poisson'] = pp
df.plot()
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e66fb6ac8>

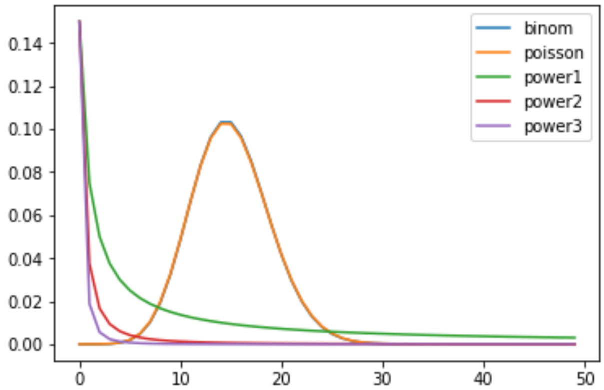


In [7]:
```
def power_dist(c, gamma, k):
    return c * k**(-gamma)
```

In [8]:
```
c = 0.15
po1 = [power_dist(c, 1, k) for k in range(1,mm+1)]
po2 = [power_dist(c, 2, k) for k in range(1,mm+1)]
po3 = [power_dist(c, 3, k) for k in range(1,mm+1)]
```

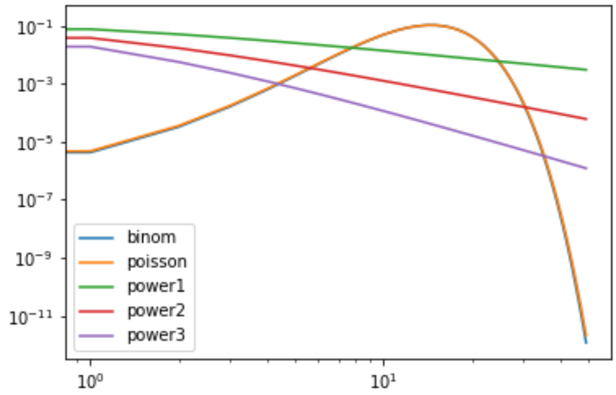
In [9]:
```
df['power1'] = po1
df['power2'] = po2
df['power3'] = po3
df.plot()
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e66c67860>



In [10]:
```
df.plot(loglog=True)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e66b9f128>



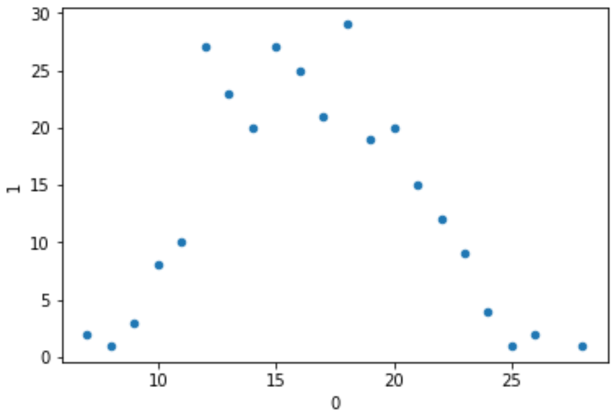
In [11]:
```
G = nx.read_pajek("c_elegans_undir.net")
G = nx.Graph(G)
```

In [12]:
```
n, m = G.number_of_nodes(), G.number_of_edges()
```

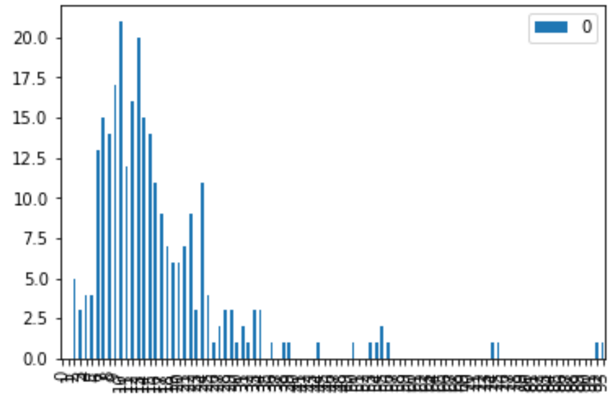
A random graph `R` of same degree n and size m.

In [13]:
```
R = nx.gnm_random_graph(n, m)
hist = nx.degree_histogram(R)
hist = [(i, hist[i]) for i in range(len(hist)) if hist[i] > 0]
df = pd.DataFrame(hist)
df.plot.scatter(x = 0, y = 1)
```

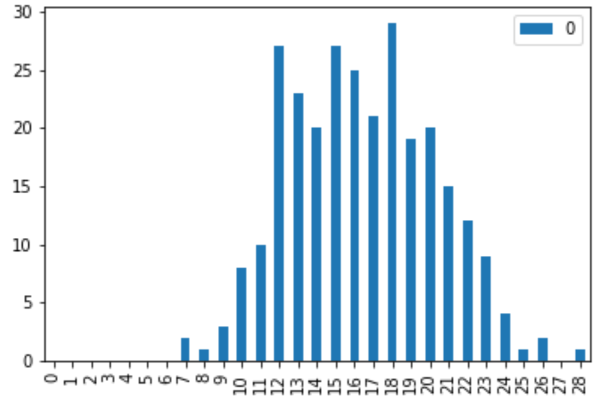
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e66a57128>



In [14]:
```
pd.DataFrame(nx.degree_histogram(G)).plot.bar()
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e66921198>



In [15]:
```
pd.DataFrame(nx.degree_histogram(R)).plot.bar()
```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e6666d5f8>



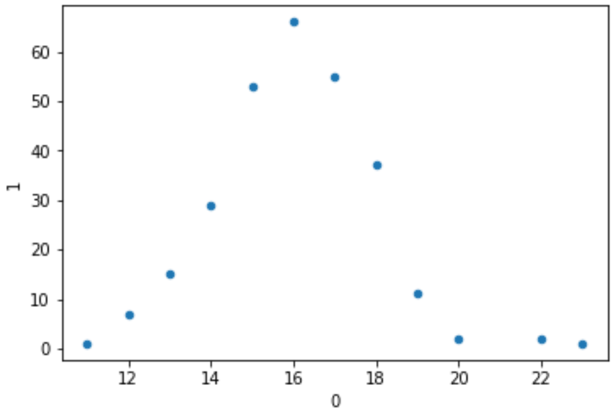
A (n, d, p)-Watts-Strogatz graph has n nodes and dn edges

In [16]:
```
d = m//n
p = 0.2
W = nx.watts_strogatz_graph(n, 2*d, p)
```

In [17]:
```
W.number_of_nodes(), W.number_of_edges()
```
Out[17]: (279, 2232)

In [18]:
```
hist = nx.degree_histogram(W)
hist = [(i, hist[i]) for i in range(len(hist)) if hist[i] > 0]
df = pd.DataFrame(hist)
df.plot.scatter(x = 0, y = 1)
```

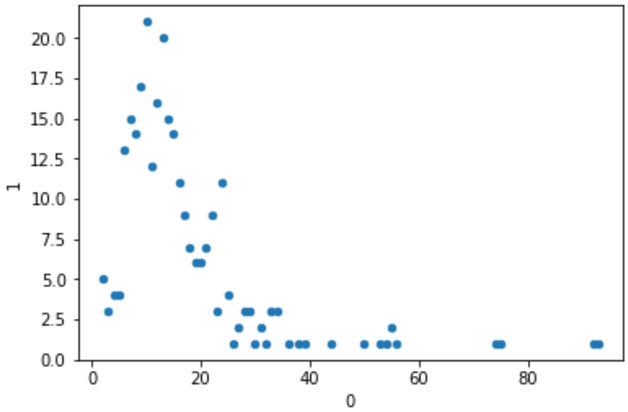
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e6699feb8>



Does the degree histogram of the worm brain network follow a power law degree distribution? Here is a standard plot and a loglog plot of it ...

In [19]:
```
hist = nx.degree_histogram(G)
hist = [(i, hist[i]) for i in range(len(hist)) if hist[i] > 0]
df = pd.DataFrame(hist)
df.plot.scatter(x = 0, y = 1)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e664a0588>



In [20]:
```
df.plot.scatter(x = 0, y = 1, loglog=True)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e6699f278>