Case Study 6 - Social Network Analysis

July 3, 2017

Basics of NetworkX

```
In [1]: import networkx as nx
```

```
In [2]: G = nx.Graph()
        G.add_node(1)
        G.add_nodes_from([2, 3])
        G.add_nodes_from(["u", "v"])
        G.nodes()
```

```
Out[2]: ['v', 1, 2, 3, 'u']
```

```
In [5]: G.add_edge(1, 2)
        G.add_edge("u", "v")
        G.add_edges_from([(1, 3), (1,4), (1,5), (1,6), ("u", "w")])
        G.edges()
```

```
Out[5]: [('w', 'u'), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), ('v', 'u')]
```

```
In [6]: G.remove_node(2)
        G.remove_nodes_from([4, 5])
        G.nodes()
```

```
Out[6]: ['w', 1, 3, 6, 'v', 'u']
```

```
In [7]: G.remove_edge(1, 3)
        G.remove_edges_from([(1, 2), ("u", "v")])
        G.edges()
```

```
Out[7]: [('w', 'u'), (1, 6)]
```

```
In [8]: G.number_of_edges()
```

```
Out[8]: 2
```

```
In [9]: G.number_of_nodes()
```

```
Out[9]: 6
```

use networkx to visualize a graph

```
In [11]: %matplotlib inline
         import matplotlib.pyplot as plt

         G = nx.karate_club_graph()
         nx.draw(G, with_labels=True, node_color="lightblue", edge_color="gray")
         plt.savefig("karate_graph.pdf")
```


```
In [12]: G.degree()

Out[12]: {0: 16,
          1: 9,
          2: 10,
          3: 6,
          4: 3,
          5: 4,
          6: 4,
          7: 4,
          8: 5,
          9: 2,
          10: 3,
          11: 1,
          12: 2,
```

```
 13: 5,
 14: 2,
 15: 2,
 16: 2,
 17: 2,
 18: 2,
 19: 3,
 20: 2,
 21: 2,
 22: 2,
 23: 5,
 24: 3,
 25: 3,
 26: 2,
 27: 4,
 28: 3,
 29: 4,
 30: 4,
 31: 6,
 32: 12,
 33: 17}
```

```
In [13]: G.degree()[10]
```

```
Out[13]: 3
```

```
In [14]: G.degree(33)
```

```
Out[14]: 17
```

build an Erdős-Rényi graph

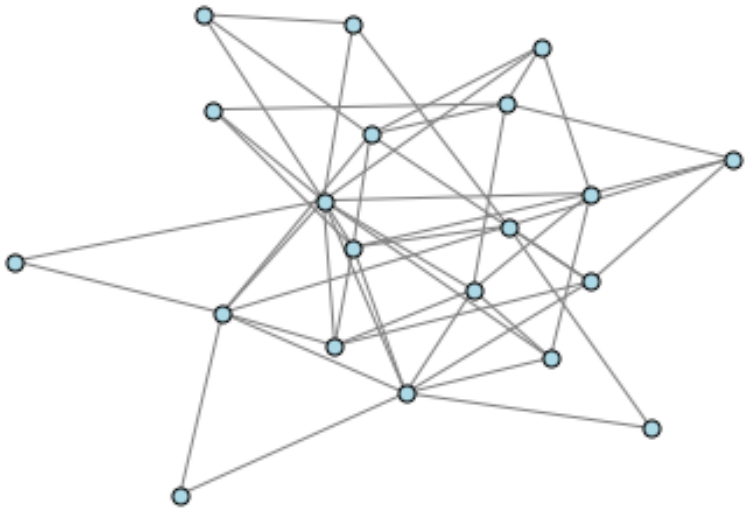
```
In [24]: from scipy.stats import bernoulli
```

```
In [28]: bernoulli.rvs(p=0.7)
```

```
Out[28]: 1
```

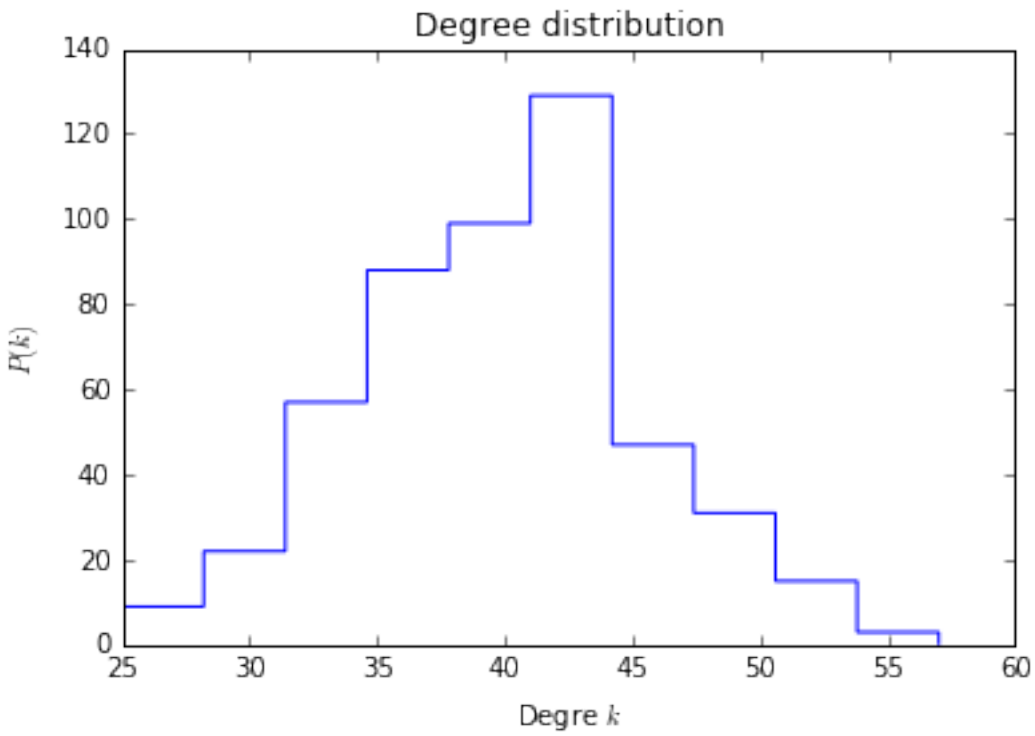
```
In [47]: def er_graph(N, p):
             """Generate an ER graph"""
             # create empty graph
             G = nx.Graph()
             # add all N nodes in the graph
             G.add_nodes_from(range(N))
             # loop over all pairs of nodes
             for node1 in G.nodes():
                 for node2 in G.nodes():
                     # add an edge with prob p
                     if node1 < node2 and bernoulli.rvs(p=p):
                         G.add_edge(node1, node2)
             return G
```

```
In [51]: nx.draw(er_graph(20, 0.2), node_size=50, node_color="lightblue", edge_colo
         plt.savefig("er1.pdf")
```

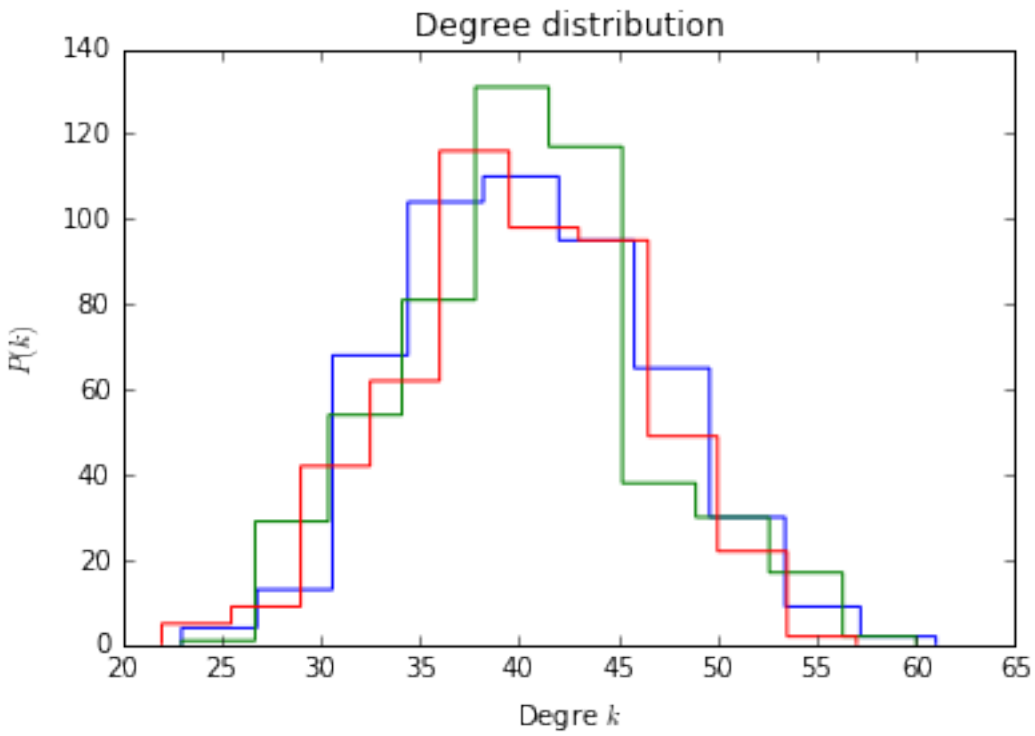


Plotting the Degree Distribution

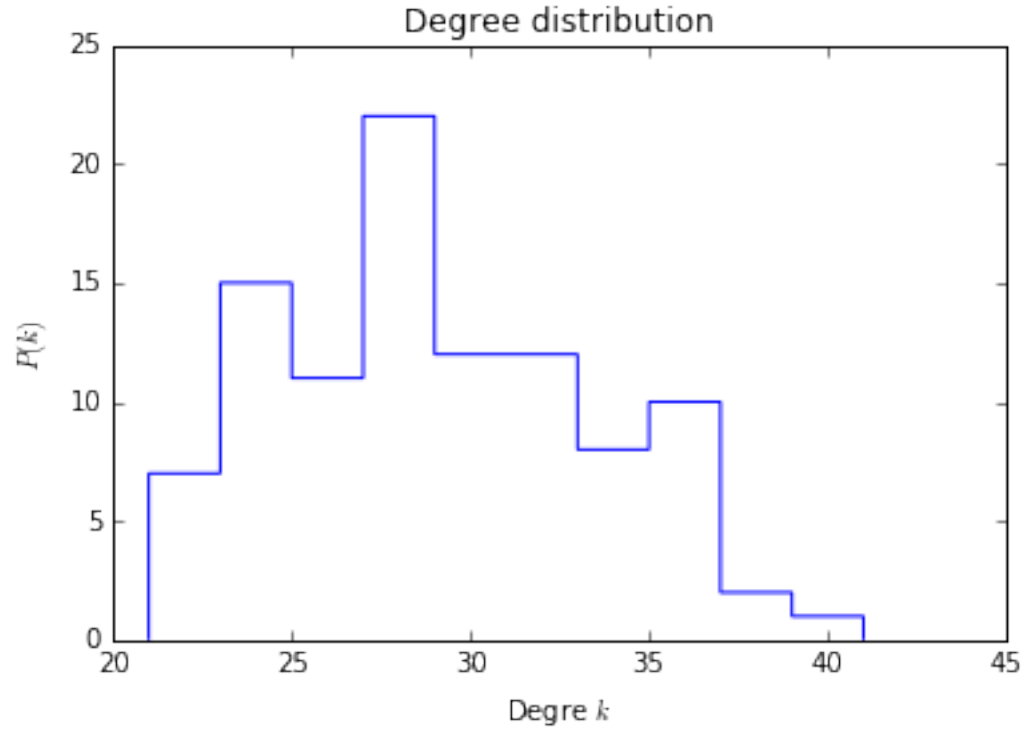
```
In [53]: def plot_degree_distribution(G):
             plt.hist(list(G.degree().values()), histtype="step")
             plt.xlabel("Degre $k$")
             plt.ylabel("$P(k)$")
             plt.title("Degree distribution")

         G = er_graph(500, 0.08)
         plot_degree_distribution(G)
         plt.savefig("hist1.pdf")
```




```
In [54]: G1 = er_graph(500, 0.08)
         plot_degree_distribution(G1)
         G2 = er_graph(500, 0.08)
         plot_degree_distribution(G2)
         G3 = er_graph(500, 0.08)
         plot_degree_distribution(G3)
         plt.savefig("hist_3.pdf")
```

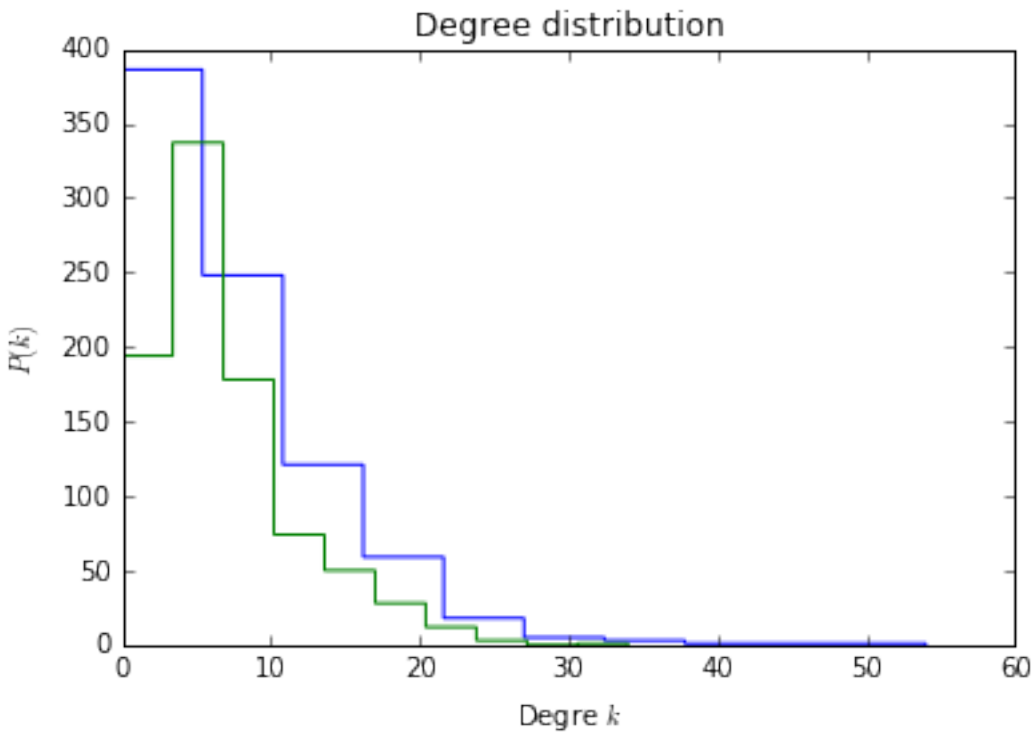



```
In [59]: plot_degree_distribution(nx.erdos_renyi_graph(100, 0.3))
```



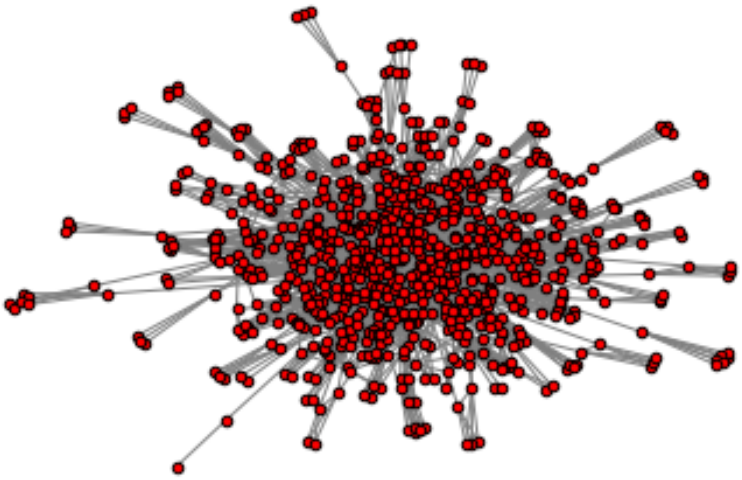
Descriptive Statistics of Empirical Social Networks

```
In [60]: import numpy as np

         A1 = np.loadtxt("adj_allVillageRelationships_vilno_1.csv", delimiter=",")
         A2 = np.loadtxt("adj_allVillageRelationships_vilno_2.csv", delimiter=",")

In [61]: G1 = nx.to_networkx_graph(A1)
         G2 = nx.to_networkx_graph(A2)

In [62]: def basic_net_stats(G):
             print("Number of nodes: %d" % G.number_of_nodes())
             print("Number of edges: %d" % G.number_of_edges())
             print("Average degree: %.2f" % np.mean(list(G.degree().values())))

         basic_net_stats(G1)
         basic_net_stats(G2)

Number of nodes: 843
Number of edges: 3405
Average degree: 8.08
Number of nodes: 877
Number of edges: 3063
Average degree: 6.99


In [63]: plot_degree_distribution(G1)
         plot_degree_distribution(G2)
         plt.savefig("village_hist.pdf")
```



Finding the Largest Connected Component

```
In [64]: G1_LCC = max(nx.connected_component_subgraphs(G1), key=len)
         G2_LCC = max(nx.connected_component_subgraphs(G2), key=len)

In [66]: len(G1_LCC)

Out[66]: 825

In [67]: len(G2_LCC)

Out[67]: 810

In [68]: len(G1_LCC) / G1.number_of_nodes()

Out[68]: 0.9786476868327402

In [69]: len(G2_LCC) / G2.number_of_nodes()

Out[69]: 0.9236031927023945

In [70]: plt.figure()
         nx.draw(G1_LCC, node_color="red", edge_color="gray", node_size=20)
         plt.savefig("village1.pdf")
```



```
In [72]: plt.figure()
         nx.draw(G2_LCC, node_color="green", edge_color="gray", node_size=20)
         plt.savefig("village2.pdf")
```