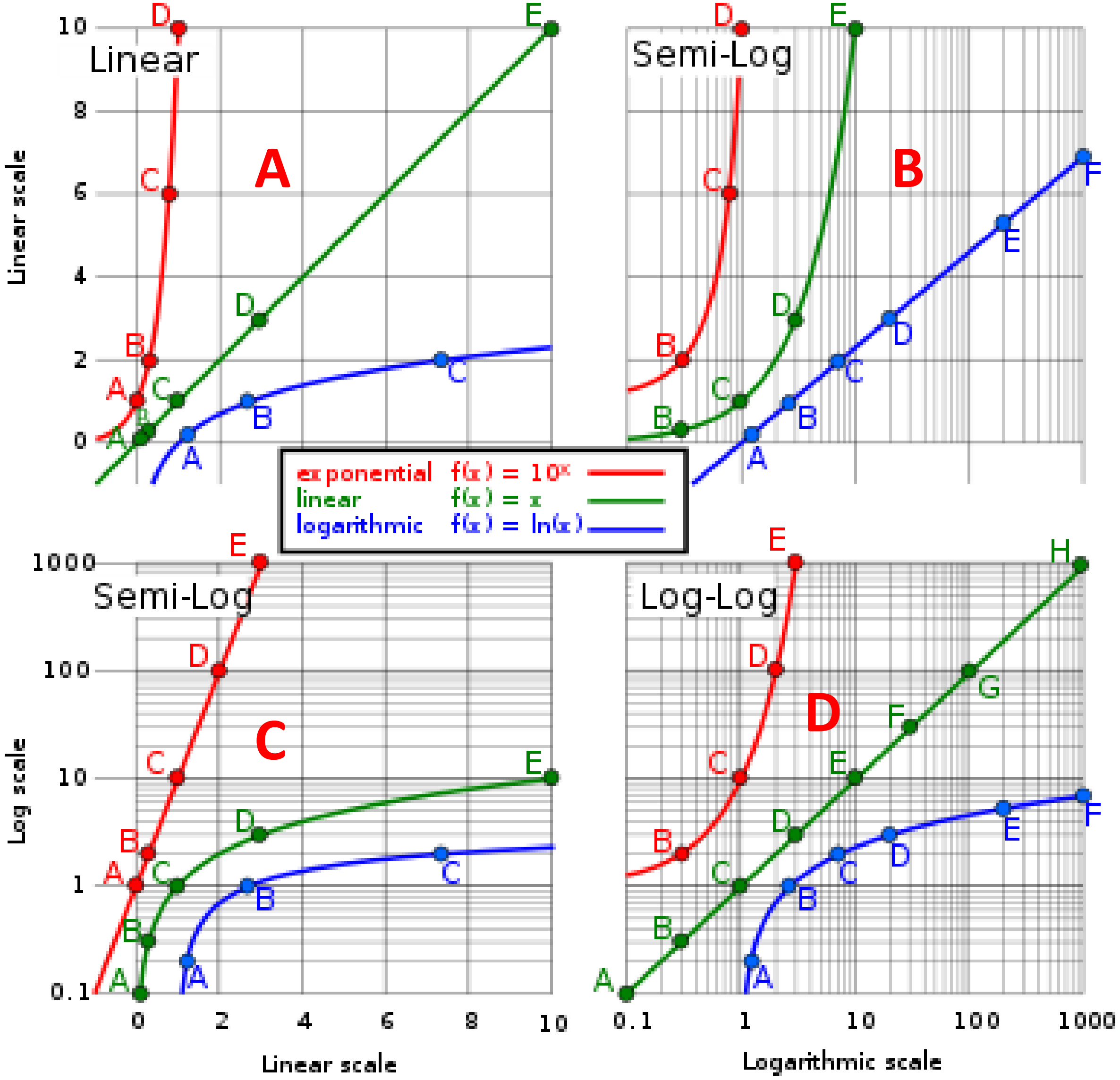
COMP 4641 Social Information Network Analysis and Engineering

Tutorial 2: Assignment 1

Yunhao GOU
ygou@connect.ust.hk

Runsheng YU
ryuah@connect.ust.hk

Logarithmic Scale

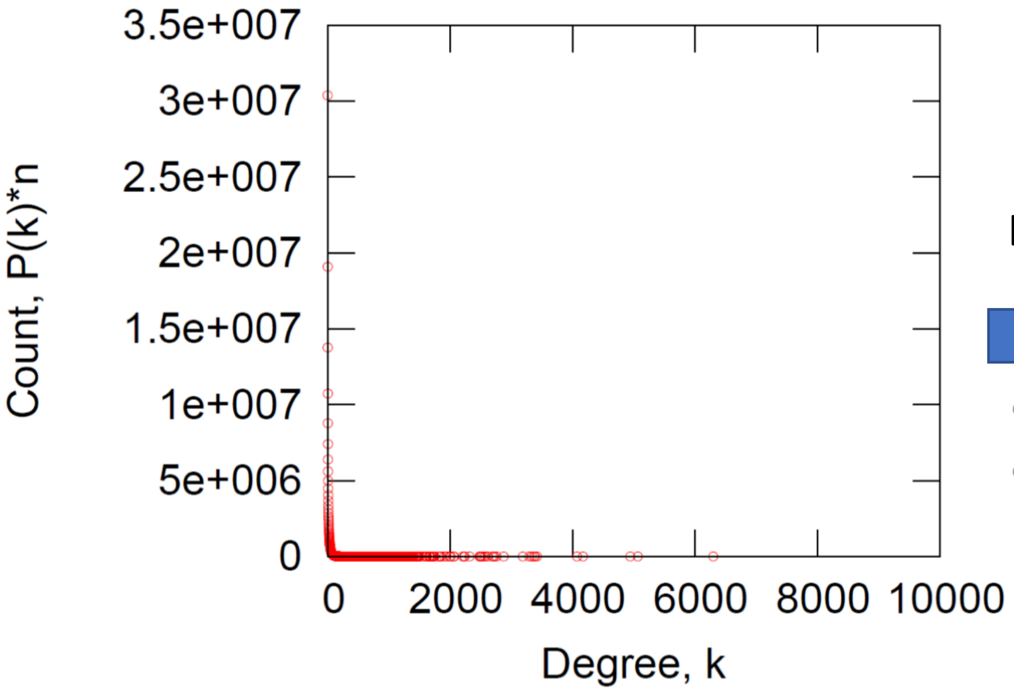


$y' = \log(y)$

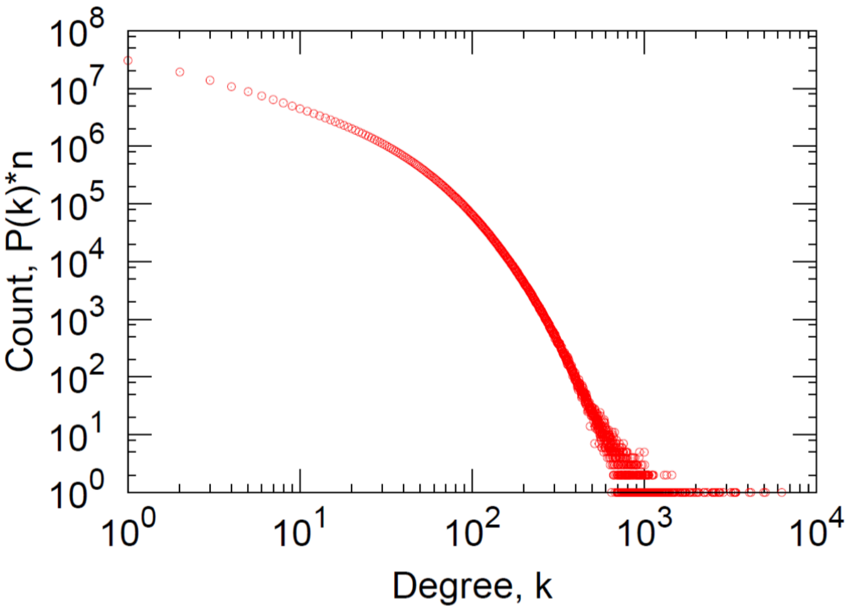
```
import matplotlib.pyplot as plt

plt.plot(x, y)          # normal ploy
plt.semilogx(x, y)      # log-scale on x axis
plt.semilogy(x, y)      # log-scale on y axis
plt.loglog(x, y)        # log-log scale
```

Image source:
https://en.wikipedia.org/wiki/Logarithmic_scale

Log-Log Scale



Log-log



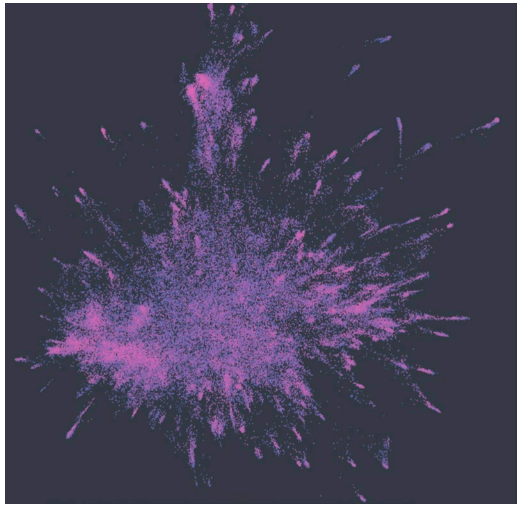
We plot the same data as on the previous slide, just the axes are now logarithmic.

Many points (small k, small count),
They are cluttered on the origin because we also have to display points that are (large k, large count)

By log-log scale
We somehow bring them together so that they could be fit in the graph

Q1&Q2

1. Use the function nx.read_edgelist(filepath, delimeter=',') to load the dataset government_edges.txt as G.
   - Rename government_edges.csv as government_edges.txt before loading.
   - Output (i) the number of nodes and edges of G; and (ii) the average degree of G.
   - Show visualization of the graph using the online tool: https://cosmograph.app/.

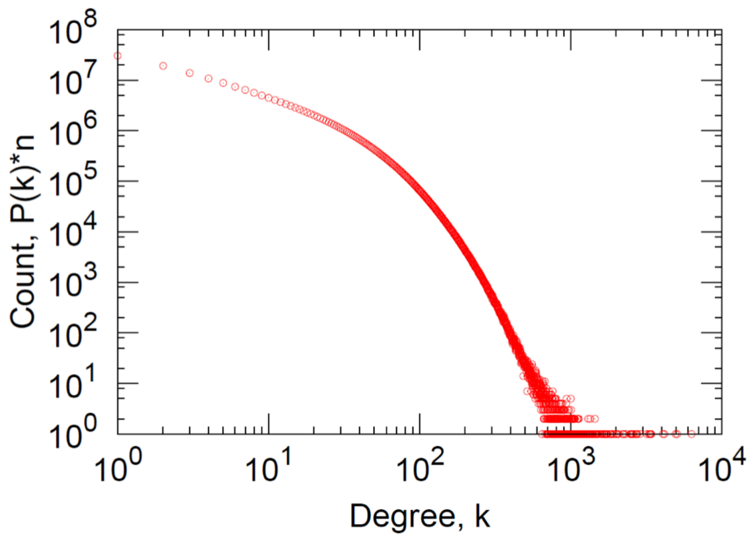


2. Use nx.gnp_random_graph() to generate a undirected graph G_{np} such that (i) the number of nodes is the same as that of G; and (ii) the expected mean degree is the same as the average degree as G.
   - Output (i) the number of nodes and edges of G_{np}; and (ii) the average degree of G_{np}.
   - Write G_{np} to file by nx.write_edgelist(Gnp, "random.csv", data=False). Visualize it using the online tool: https://cosmograph.app/.

Q3

3. Plot the degree histograms (as on **p11** of Lecture_4_SmallWorld.pdf) of G and G_{np}, using both linear-linear and log-log scales on the axes.

- Compare the histograms, what conclusions can be drawn?



Use python built-in class “Counter”

```
Counter(['a', 'b', 'c', 'b', 'b', 'b'])
```

```
Counter({'a': 1, 'b': 4, 'c': 1})
```

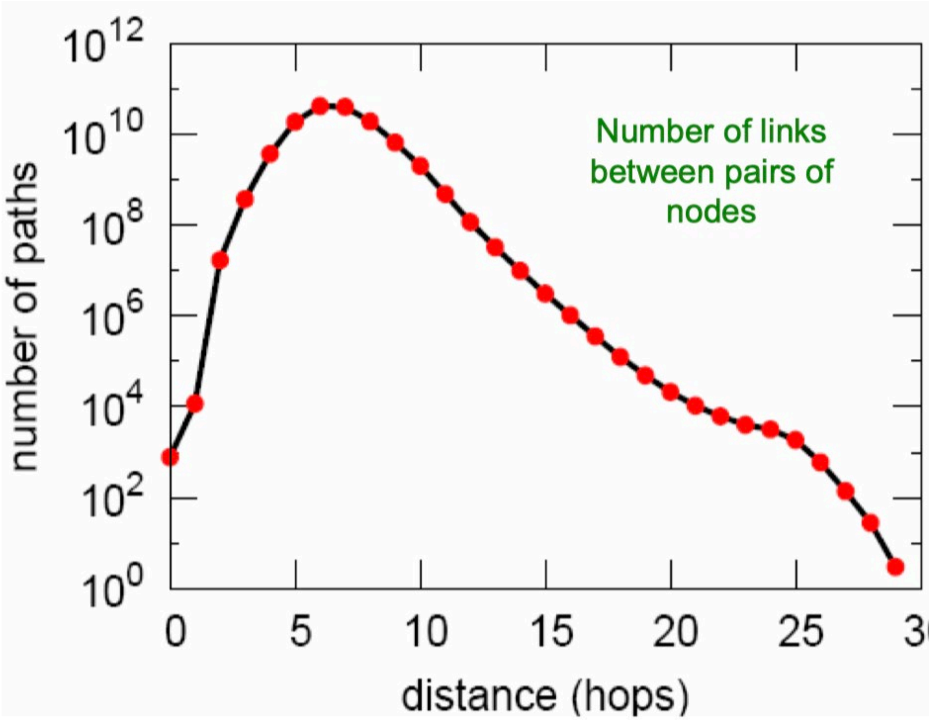
from collections import Counter

```
degree_seq  = [d for n, d in G.degree()]
# Use Counter to count distinct values in degree_seq
# Obtain the counts for all possible degrees in degree_seq
# ( from 0 to max(degree_seq) ) as degree_freq
# Do the plotting with degree_freq
```

Q4

4. Plot the number of paths versus the (shortest) path distance (as in **p19** of Lecture_4_SmallWorld.pdf) for G and G_{np}.

- From the plots obtained, do G and G_{np} have similar average path lengths?




```
dis_seq = []
path_len = nx.all_pairs_shortest_path_length(G)
for … in path_len:
    # append to dis_seq

# count the distinct values in dis_seq
# Obtain the counts for all possible distances in dis_seq
# Do the plotting
```

path_len:

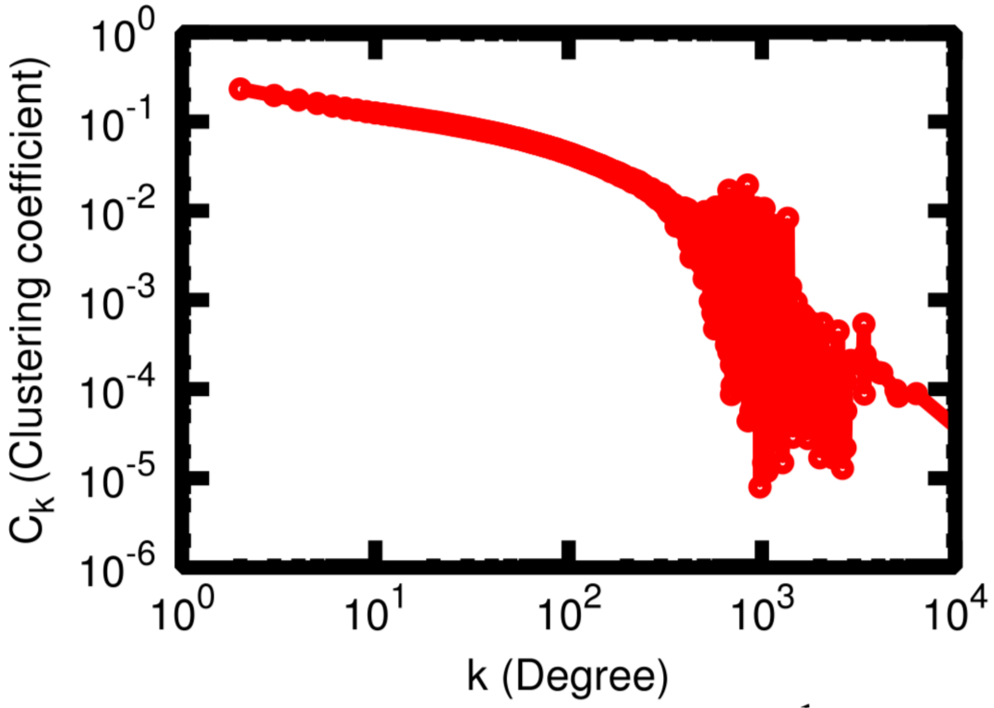
(source1, source_to_target_dict)
(source2, source_to_target_dict)

source_to_target_dict:
{'target1': 3, 'target2': 4, ….., }

Q5

5. Plot the average clustering coefficient versus degree (as in **p26** of Lecture_4_SmallWorld.pdf) for G and G_{np}

- Both axes should be in log scale.



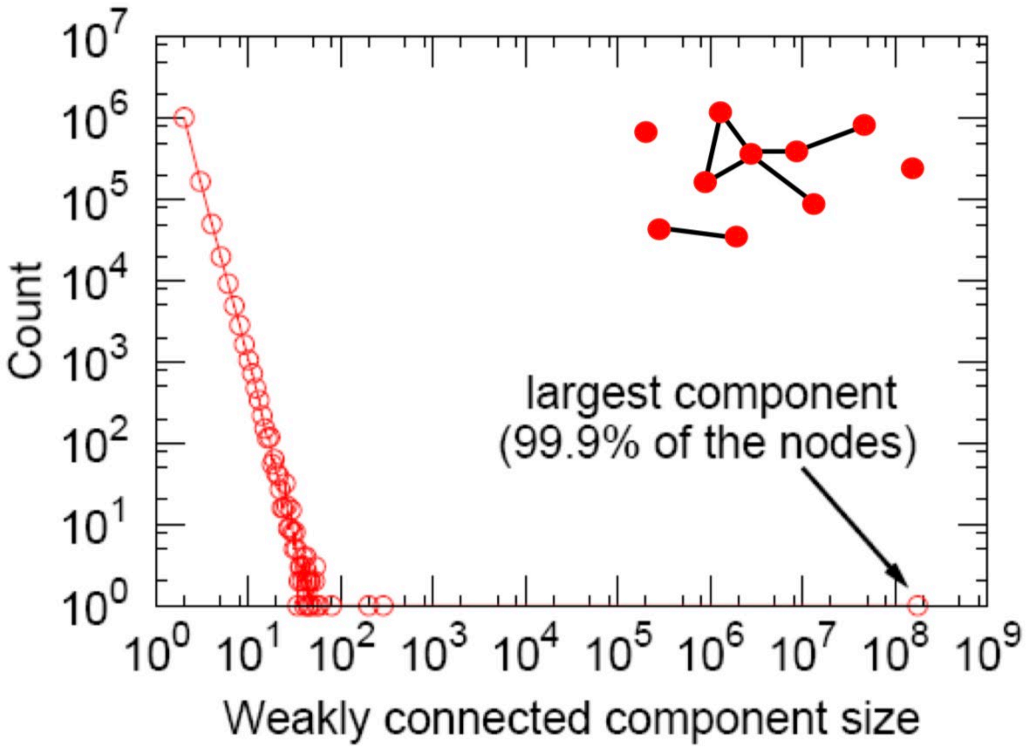
C_k: average C_i of nodes i of degree k: $C_k = \frac{1}{N_k} \sum_{i:k_i=k} C_i$

```
# Obtain the degree sequence (degree_seq)  [d1, d2, ..., dN]
# Obtain clustering coefficient for all the nodes (coef)  [c1, c2, ..., cN]

coef_d_list = []
For d in range(max(degree_seq):
        # find a mask for degree=d in degree_seq
        # apply the mask on coef and compute the mean
        # add to coef_d_list

# Do the plotting
```

Q6

6. Plot the number of connected components with size (as in **p30** of Lecture_4_SmallWorld.pdf).

- From the plots obtained, what do G and G_{np} have in common?



\# Obtain the size of all connected components (comp_sizes)

```
w_comp_size = [len(comp) for comp in func(G)]
```

\# Count the distinct values of comp_sizes using Counter
\# Obtain the counts for all possible sizes in comp_sizes
\# Do the plotting