The Big Picture: Describing Networks

```python
# Configure plotting in Jupyter
from matplotlib import pyplot as plt
%matplotlib inline
plt.rcParams.update({
    'figure.figsize': (7.5, 7.5)})
# Seed random number generator
from numpy import random as nprand
seed = hash("Network Science in Python") % 2**32
nprand.seed(seed)
```

```python
import networkx as nx
```

Data Sets

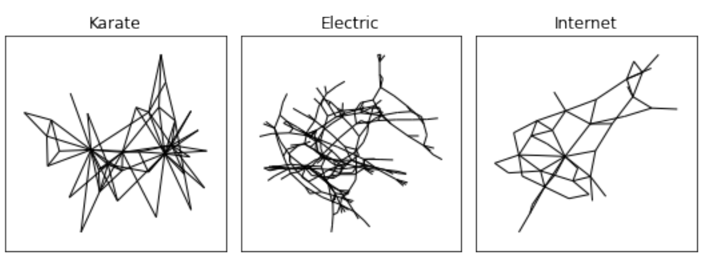
```python
from pathlib import Path
data_dir = Path('.') / 'data'
```

```python
# Load karate club network
G_karate = nx.karate_club_graph()
mr_hi = 0
john_a = 33
```

```python
# Load internet point of presence network
G_internet = nx.read_graphml(data_dir / 'UAITZ' / 'Geant2012.graphml')
```

```python
# Load Germany electrical grid
with open(data_dir / 'mureddu2016' / '0.2' / 'branches.csv', 'rb') as f:
    # Skip header
    next(f)
    # Read edgelist format
    G_electric = nx.read_edgelist(
        f,
        delimiter="\t",
        create_using=nx.Graph,
        data=[('X', float), ('Pmax', float)])
```

```python
plt.figure(figsize=(7.5, 2.75))
plt.subplot(1, 3, 1);
plt.title("Karate")
nx.draw_networkx(G_karate, node_size=0, with_labels=False)
plt.subplot(1, 3, 2)
plt.title("Electric")
nx.draw_networkx(G_electric, node_size=0, with_labels=False)
plt.subplot(1, 3, 3)
plt.title("Internet")
nx.draw_networkx(G_internet, node_size=0, with_labels=False)
plt.tight_layout()
```



Diameter and Shortest Paths

```python
list(nx.all_shortest_paths(G_karate, mr_hi, john_a))
```

```
[[0, 8, 33], [0, 13, 33], [0, 19, 33], [0, 31, 33]]
```

```python
nx.shortest_path(G_karate, mr_hi, john_a)
```

```
[0, 8, 33]
```

```python
nx.shortest_path_length(G_karate, mr_hi, john_a)
```

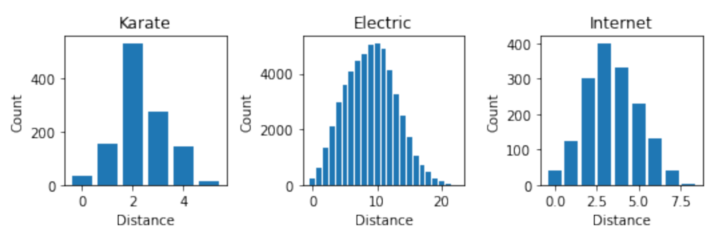
```
2
```

```python
# Calculate dictionary of all shortest paths
length_source_target = dict(nx.shortest_path_length(G_karate))
length_source_target[0][33]
```

```
2
```

```python
def path_length_histogram(G, title=None):
    # Find path lengths
    length_source_target = dict(nx.shortest_path_length(G))
    # Convert dict of dicts to flat list
    all_shortest = sum(
        [list(length_target.values()) for length_target in length_source_target.values()],
        [])
    # Calculate integer bins
    high = max(all_shortest)
    bins = [-0.5 + i for i in range(high + 2)]
    # Plot histogram
    plt.hist(all_shortest, bins=bins, rwidth=0.8)
    plt.title(title)
    plt.xlabel("Distance")
    plt.ylabel("Count")
```

```python
plt.figure(figsize=(7.5, 2.5))
plt.subplot(1, 3, 1)
path_length_histogram(G_karate, title="Karate")
plt.subplot(1, 3, 2)
path_length_histogram(G_electric, title="Electric")
plt.subplot(1, 3, 3)
path_length_histogram(G_internet, title="Internet")
plt.tight_layout()
```




```python
nx.average_shortest_path_length(G_karate)
```

```
2.408199643493761
```

```python
nx.average_shortest_path_length(G_electric)
```

```
9.044193487671748
```

```python
nx.average_shortest_path_length(G_internet)
```

```
3.528205128205128
```

```python
nx.diameter(G_karate)
```

```
5
```

```python
nx.diameter(G_electric)
```

```
22
```

```python
nx.diameter(G_internet)
```

```
8
```

Clustering

```python
nx.transitivity(G_karate)
```

```
0.2556818181818182
```

```python
nx.transitivity(G_electric)
```

```
0.07190412782956059
```

```python
nx.transitivity(G_internet)
```

```
0.135678391959799
```

```python
nx.average_clustering(G_karate)
```

```
0.5706384782076823
```

```python
nx.average_clustering(G_electric)
```

```
0.06963512677798392
```

```python
nx.average_clustering(G_internet)
```

```
0.1544047619047619
```

Resilience

```python
nx.density(G_karate)
```

```
0.13903743315508021
```

```python
nx.density(G_electric)
```

```
0.011368341803124411
```

```python
nx.density(G_internet)
```

```
0.0782051282051282
```

```python
import networkx.algorithms.connectivity as nxcon
```

```python
nxcon.minimum_st_node_cut(G_karate, mr_hi, john_a)
```

```
{2, 8, 13, 19, 30, 31}
```

```python
nxcon.minimum_st_edge_cut(G_karate, mr_hi, john_a)
```

```
{(0, 8),
 (0, 31),
 (1, 30),
 (2, 8),
 (2, 27),
 (2, 28),
 (2, 32),
 (9, 33),
 (13, 33),
 (19, 33)}
```

```python
nx.node_connectivity(G_karate, mr_hi, john_a)
```

```
6
```

```python
nx.edge_connectivity(G_karate, mr_hi, john_a)
```

```
10
```

```python
nxcon.minimum_node_cut(G_karate)
```

```
{0}
```

```python
nxcon.minimum_edge_cut(G_karate)
```

```
{(11, 0)}
```

```python
nx.node_connectivity(G_karate)
```

```
1
```

```python
nx.node_connectivity(G_electric)
```

```
1
```

```python
nx.node_connectivity(G_internet)
```

```
1
```

```python
nx.average_node_connectivity(G_karate)
```

```
2.2174688057040997
```

```python
nx.average_node_connectivity(G_electric)
```

```
1.5188029361942406
```

```python
nx.average_node_connectivity(G_internet)
```

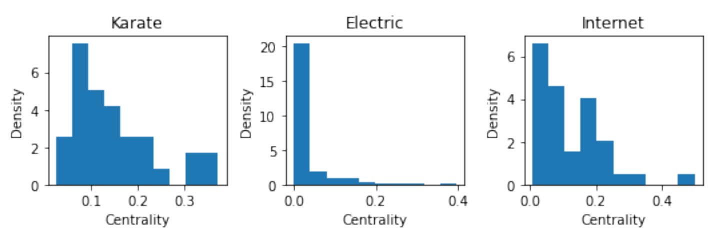
```
1.7346153846153847
```

Inequality

```python
# Function to plot a single histogram
def centrality_histogram(x, title=None):
    plt.hist(x, density=True)
    plt.title(title)
    plt.xlabel("Centrality")
    plt.ylabel("Density")

# Create a figure
plt.figure(figsize=(7.5, 2.5))
# Calculate centralities for each example and plot
plt.subplot(1, 3, 1)
centrality_histogram(
    nx.eigenvector_centrality(G_karate).values(), title="Karate")
plt.subplot(1, 3, 2)
centrality_histogram(
    nx.eigenvector_centrality(G_electric, max_iter=1000).values(), title="Electric")
plt.subplot(1, 3, 3)
centrality_histogram(
    nx.eigenvector_centrality(G_internet).values(), title="Internet")

# Adjust the layout
plt.tight_layout()
```




```python
import math
def entropy(x):
    # Normalize
    total = sum(x)
    x = [xi / total for xi in x]
    H = sum([-xi * math.log2(xi) for xi in x])
    return H
```

```python
entropy(nx.eigenvector_centrality(G_karate).values())
```

```
4.842401948329853
```

```python
entropy(nx.eigenvector_centrality(G_electric, max_iter=1000).values())
```

```
6.030447144924192
```

```python
entropy(nx.eigenvector_centrality(G_internet).values())
```

```
4.86203726163741
```

```python
def gini(x):
    x = [xi for xi in x]
    n = len(x)
    gini_num = sum([sum([abs(x_i - x_j) for x_j in x]) for x_i in x])
    gini_den = 2.0 * n * sum(x)
    return gini_num / gini_den
```

```python
gini(nx.eigenvector_centrality(G_karate).values())
```

```
0.3244949051532847
```

```python
gini(nx.eigenvector_centrality(G_electric, max_iter=1000).values())
```

```
0.787950636595495
```

```python
gini(nx.eigenvector_centrality(G_internet).values())
```

```
0.4343286009726223
```

```python
nx
```

```
<module 'networkx' from '/Users/NathanBick/miniconda3/envs/network/lib/python3.10/site-packages/networkx/__init__.py'>
```