RamaContPaper

February 6, 2024

1 Network Demand Dynamics: An Erdős–Rényi Model Analysis

1.1 Abstract

This study investigates the distribution of aggregate demand in an Erdős–Rényi random graph model with 1000 agents. By simulating the network 1000 times, I examine the effects of varying network density on the demand distribution, characterized by the parameter `c`.

1.2 Methodology

The network is modeled with a fixed number of agents (N=1000) where each agent has a demand state of 1, -1, or 0, with the respective probabilities determined by `a=0.2`. Each iteration generates a new network instance, and the total demand is measured across connected components.

1.3 Results

The distributions of summed demands for various `c` values were recorded, with kurtosis calculated to assess the "tailedness" of the distribution. Variance and the fourth moment provided additional insights into the distribution's dispersion and peakness.

1.4 Discussion

my analysis indicates that denser networks (higher `c`) significantly affect kurtosis, suggesting a link between network structure and systemic risk. These findings can inform risk mitigation strategies in complex financial systems.

1.5 Conclusion

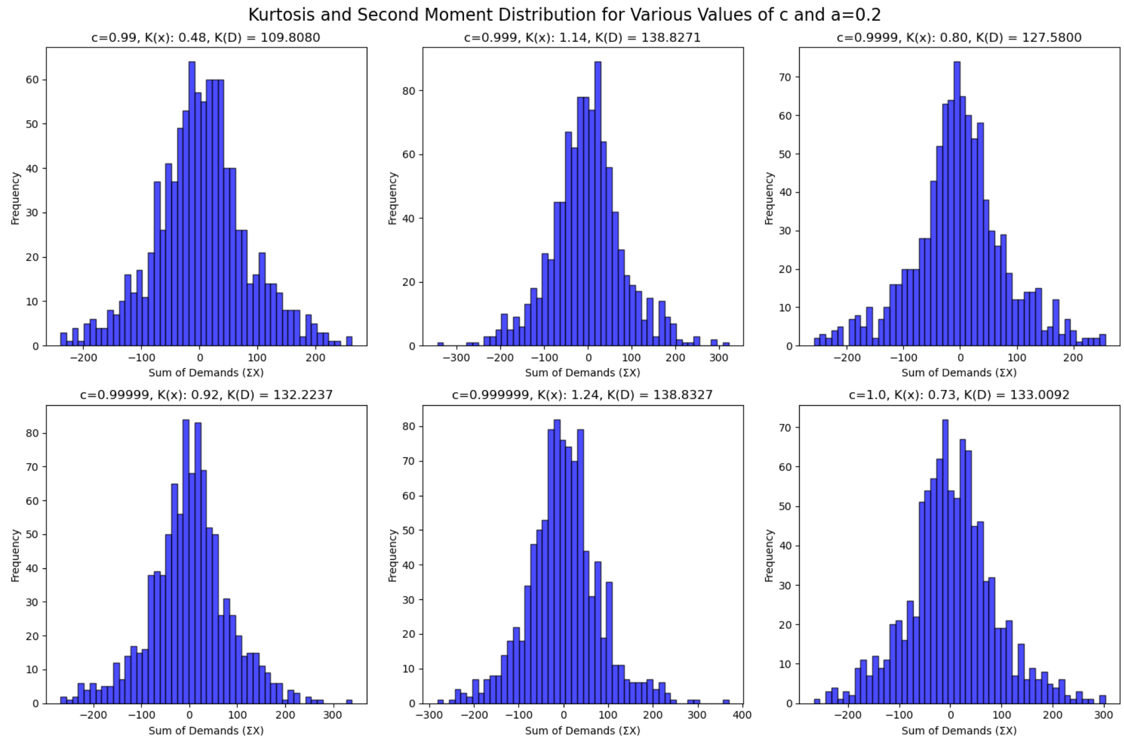
The study underscores the intricate relationship between network density and the distribution of demand, highlighting the potential systemic risks in densely interconnected networks. In my simulation, I did not obseve a phase transition around `c`=1.

[1]:
```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from scipy.stats import kurtosis

# Parameters
N = 1000  # Number of agents
```

```python
cs = [0.99, 0.999, 0.9999, 0.99999, 0.999999, 0.999999999999999999]  # Updated to have exactly 6 values
a = 0.2   # Probability for 1 and -1
iterations = 1000  # Number of iterations to repeat the experiment
N_order=2*a*N
# Set up a 2x3 subplot grid
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
axes = axes.flatten()  # Flatten the 2D array of axes for easy iteration

# Loop over each value of c and create a plot for each
for index, c in enumerate(cs):
    sum_of_demands = []  # Reset the storage for sum of demands

    for _ in range(iterations):
        # Create an Erdős-Rényi network G(N, p)
        p = c / (N - 1)
        G = nx.erdos_renyi_graph(N, p)

        # Determine the clusters and calculate demands
        total_demand = sum(
            len(cluster) * np.random.choice([1, -1, 0], p=[a, a, 1 - 2 * a])
            for cluster in nx.connected_components(G)
        )

        # Store the sum of demands for this iteration
        sum_of_demands.append(total_demand)

    # Calculate kurtosis for the current c value
    k_x = kurtosis(sum_of_demands, fisher=True)  # K(x): Excess kurtosis
    mu_2_Xa = np.var(sum_of_demands)  # Second moment, variance of the demands
    mu_4_Xa = np.mean((sum_of_demands - np.mean(sum_of_demands))**4)  # Fourth moment
    k_d = mu_4_Xa / (N_order * (1 - c / 2) * mu_2_Xa)

    # Plot the distribution of the sum of demands for the current value of c
    axes[index].hist(sum_of_demands, bins=50, alpha=0.7, color='blue', edgecolor='black')
    axes[index].set_title(f'c={c}, K(x): {k_x:.2f}, K(D) = {k_d:.4f}')  # Format kurtosis to 2 decimal places
    axes[index].set_xlabel('Sum of Demands (ΣX)')
    axes[index].set_ylabel('Frequency')
    print(f"The kurtosis of the distribution for c={c} is: {k_x:.2f}, K(D) = {k_d:.4f}")  # Print kurtosis to 2 decimal places

# Adjust layout to prevent overlap
fig.suptitle(f'Kurtosis and Second Moment Distribution for Various Values of c and a={a}', fontsize=16)
```

```
plt.tight_layout()
plt.show()
```

```
The kurtosis of the distribution for c=0.99 is: 0.48, K(D) = 109.8080
The kurtosis of the distribution for c=0.999 is: 1.14, K(D) = 138.8271
The kurtosis of the distribution for c=0.9999 is: 0.80, K(D) = 127.5800
The kurtosis of the distribution for c=0.99999 is: 0.92, K(D) = 132.2237
The kurtosis of the distribution for c=0.999999 is: 1.24, K(D) = 138.8327
The kurtosis of the distribution for c=1.0 is: 0.73, K(D) = 133.0092
```



[ ]:

```

```