# ES 404: Networks and Complex Systems

# Maritime Port Connectivity Network

**Shiv Nitinkumar Patel - 23110302**

---

Link to the assignment GitHub repo: link

## Task 1: Network Construction & Preliminaries

1. **Dataset Description:**
   Provide context about your network:
   I have chosen my project to be "**Maritime Port Connectivity Network"**. For this Assignment, I am using my Own Project Dataset – **PLSBCI** (Port Linear Shipping Bilateral Connectivity Index).
   The network represents maritime connectivity between global container ports, based on a combination of port-level and country-level shipping indices. This network was constructed as part of a project using data from the **UNCTAD PLSCI and LSBCI indices**.

- What does each node represent?

**Ans** - Each **node** represents a **port**.

**a.** More specifically, a node corresponds to a unique **Port Label** from the PLSCI dataset provided by **UNCTAD** (UN Trade and Development).

b. This label usually contains the **port name** and the **country** (e.g., *"Shanghai, China"*, *"Rotterdam, Netherlands"*).

c. These are **container ports** evaluated for their connectivity.

- What do the edges signify?

**Ans**- Each **edge** represents the **bilateral maritime connectivity** between **two specific ports**.

1. The **port-level connectivity (PLSCI)** of each port, and
2. The **country-level bilateral connectivity (LSBCI)** between their respective countries.

This essentially:

Enhances **port-to-port connectivity** with respect to **country-to-country trade links.**

Captures both **port performance** and **diplomatic/trade connectivity** between their nations.

- Where/how did you acquire this data?

**Ans - Data Acquisition**

**A. PLSCI** (Port Liner Shipping Connectivity Index)

**Source**: UNCTAD (United Nations Conference on Trade and Development)

**Access**: Publicly available via the UNCTAD statistics portal.

**What it contains**:

1. Individual **container ports**
2. Their **connectivity to global liner shipping networks**
3. Measured by metrics like frequency of services, number of operators, direct connections, and size of ships.

**B. LSBCI** (Liner Shipping Bilateral Connectivity Index)

**Source**: UNCTAD

**Access**: Also downloadable via UNCTAD statistics portal.

**What it contains**:

1. **Country-pair level** connectivity values.
2. Based on number of transshipments, carriers, and indirect/direct maritime links
3. Each row includes Economy_Label and Partner_Label (countries) and Index_Value (connectivity score).

2. **Construct Your Network:**

Describe any data cleaning/preprocessing steps before building the adjacency list/matrix.

Code for Data Cleaning Process: link

**Step 1:** Formation of ${}^{N}C_2$ graph.

First, I combined **port-level connectivity (PLSCI)** and **country-level bilateral connectivity (LSBCI) data** to compute a new metric called **PLSBCI** (Port Linear Shipping Bilateral Connectivity Index) for every port-pair. It first performs a **cross joint** to generate all possible port combinations, then extracts country names to match with LSBCI data. After merging, it applies to a **weighted average formula** to calculate PLSBCI. The result captures both port performance and inter-country trade strength.

Mathematically, it uses a **weighted arithmetic mean**:

$$PLSBCI = \frac{(PLSCI_A + PLSCI_B) * LSBCI_{A,B}}{2}$$

**Explanation of each term:**

- $\boldsymbol{PLSCI_A}$: Port Liner Shipping Connectivity Index of Port A — measures how well Port A is connected in global shipping.
- $\boldsymbol{PLSCI_B}$: Port Liner Shipping Connectivity Index of Port B — same as above, but for Port B.
- $\boldsymbol{LSBCI_{A,B}}$: Liner Shipping Bilateral Connectivity Index between the countries of Port A and Port B — shows how strong the shipping connection is between their countries.
- **Final PLSBCI**: This indicates the bilateral shipping connectivity index between Port A and Port B .
-

```
import pandas as pd

# Load datasets
plsc_df = pd.read_csv(r"B:\Final Network Science
Project\Data\PLSCI\PLSCI.csv")  # Port Linear Shipping Connectivity Index
lsbci_df = pd.read_csv(r"B:\Final Network Science Project\Data\LSBCI\LSBCI.csv")
# Linear Shipping Bilateral Connectivity Index

# Step 1: Cross join PLSCI dataset
plsc_df['key'] = 1
cross_joined = plsc_df.merge(plsc_df, on='key', suffixes=('_A',
'_B')).drop('key', axis=1)

# Step 2: Extract countries
cross_joined['country_A'] = cross_joined['Port_Label_A'].str.split(', ').str[0]
cross_joined['country_B'] = cross_joined['Port_Label_B'].str.split(', ').str[0]

# Step 3: Merge LSBCI data
merged = cross_joined.merge(lsbci_df, left_on=['country_A', 'country_B'],
right_on=['Economy_Label', 'Partner_Label'], how='left')
merged.rename(columns={'Index_Value': 'LSBCI_AB'}, inplace=True)

# Step 4: Apply formula
merged['PLSBCI'] = ((merged['Index_Value_A'] * merged['LSBCI_AB']) +
                    (merged['Index_Value_B'] * merged['LSBCI_AB'])) / 2
# Step 5: Select required columns
result = merged[['Port_Label_A', 'Port_Label_B', 'PLSBCI']]
```

```
result.to_csv(r"PLSBCI_Output.csv", index=False)
```

**Step 2:** Making the data realistic by applying the Threshold.

From the above code that I wrote I made a graph with all possible $^{908}C_2$ port-pair combinations i.e. 411,778, but only **12,749 port pairs** are known to have direct liner shipping connections that accounts for only **3.10%**, as reported by the UN.
So I wrote a code that filters the dataset to **retain only the top 12,749 port pairs** with the highest PLSBCI values — which are most likely to represent actual shipping routes. It does this by:

- Sorting the port pairs by their connectivity scores,
- Finding the PLSBCI value at the 12,749th highest rank.
- Keeping only those port pairs with a score above or equal to that value.
- And zeroing out all weaker or non-existent links.

This ensures that your final graph structure is **realistic.**

```
import pandas as pd
import numpy as np

# Read the CSV file
df = pd.read_csv(r'B:\Final Network Science Project\PLSBCI_Output.csv')

# Sort the dataframe by Index in descending order
df_sorted = df.sort_values('PLSBCI', ascending=False)

# Get the 25,496th highest value (which is the 12,749th pair)
threshold = df_sorted.iloc[25496]['PLSBCI']

# Modify the original Index column to keep only the top 12,749 links.
df['PLSBCI'] = np.where(df['PLSBCI'] >= threshold, df['PLSBCI'], 0)

# Save the modified dataframe to a new CSV file
df.to_csv('B:\Final Network Science Project\Data\Mr_PLSBCI.csv', index=False)
```

**Step 3:** Addition Of Coordinates

The data that I took from the UN Trade and Development didn't had Coordinates of each Port so, I opened the generated Graph of Shipping port network csv file that I processed In Step 2 and used the Geocode Extension to add the Latitude and Longitude of each and every port address.

**Step 4:** Converting into graphml file format for easy Processing of Network.

The given data that I Processed Till now was in CSV format, so now I loaded the Network data CSV file into the Gephi Software and Exported it as "**Shipping_Network.graphml**" file to have easy processing of the network.

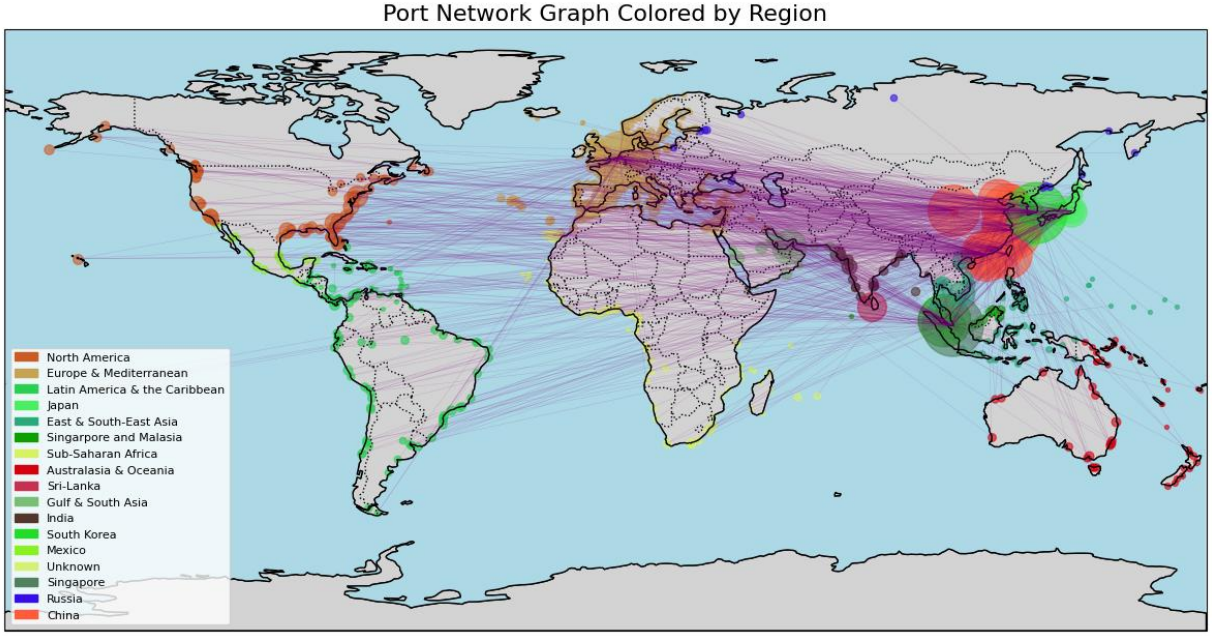
**Step 5:** Adding Region Column

I Added a column called a "Region" to represent the port data as a cluster of Countries / Continent the specific port belong to , I did this by defining the the latitude and longitude extent for each region.

```
import xml.etree.ElementTree as ET

# Function to determine the region based on latitude and longitude
def determine_region(lat, lon):
    lat = float(lat)
    lon = float(lon)

    # Special bifurcation for China, India, and Singapore
    if 18 <= lat <= 55 and 73 <= lon <= 135:  # Approximate range for China
        return "China"
    elif 6 <= lat <= 35 and 68 <= lon <= 97:  # Approximate range for India
        return "India"
    elif 1 <= lat <= 2 and 103 <= lon <= 104:  # Approximate range for Singapore
        return "Singapore"

    # General regions
    if -50 <= lat <= 0 and 110 <= lon <= 180:
        return "Australasia & Oceania"
    elif -10 <= lat <= 60 and 90 <= lon <= 180:
        return "East & South-East Asia"
    elif 30 <= lat <= 70 and -30 <= lon <= 60:
        return "Europe & Mediterranean"
    elif 0 <= lat <= 30 and 40 <= lon <= 90:
        return "Gulf & South Asia"
    elif -60 <= lat <= 30 and -120 <= lon <= -30:
        return "Latin America & the Caribbean"
    elif 20 <= lat <= 80 and -170 <= lon <= -50:
        return "North America"
    elif -35 <= lat <= 15 and -20 <= lon <= 55:
        return "Sub-Saharan Africa"
    else:
        return "Unknown"
```

```
# Parse the GraphML file
graphml_file = r"B:\Final Network Science Project\Data\grpah
file\FinalPortNetworkGrpah.graphml"
tree = ET.parse(graphml_file)
root = tree.getroot()

# Define namespace
ns = {'': 'http://graphml.graphdrawing.org/xmlns'}
ET.register_namespace('', ns[''])

# Add a new key for the region
ET.SubElement(root, 'key', {
    'id': 'region',
    'for': 'node',
    'attr.name': 'region',
    'attr.type': 'string'
})

# Iterate over nodes and assign regions
for node in root.findall('.//{http://graphml.graphdrawing.org/xmlns}node'):
    latitude = None
    longitude = None

    # Find latitude and longitude for the node
    for data in node.findall('./{http://graphml.graphdrawing.org/xmlns}data'):
        if data.get('key') == 'latitude':
            latitude = data.text
        elif data.get('key') == 'longitude':
            longitude = data.text

    # Assign region if latitude and longitude are found
    if latitude is not None and longitude is not None:
        region = determine_region(latitude, longitude)
        ET.SubElement(node, '{http://graphml.graphdrawing.org/xmlns}data',
{'key': 'region'}).text = region

# Save the modified GraphML file
output_file = r"B:\Final Network Science Project\Data\grpah
file\portnetwork.graphml"
tree.write(output_file, encoding='utf-8', xml_declaration=True)
print(f"Modified GraphML file with regions saved at: {output_file}")
```

Port Network Graph Colored by Region




Now after completing all the above 5 steps my final network graph/Matrix is ready.

Link to the complete network file : link

3. **Initial Observations:** link

- Report the number of nodes (*N*) and edges (*E*).
- Number of nodes (N) : **908**
- Number of edges (E) : **12749**

- Plot your network's **degree distribution** with proper binning. Comment on any notable patterns (e.g., skewness, heavy tail).

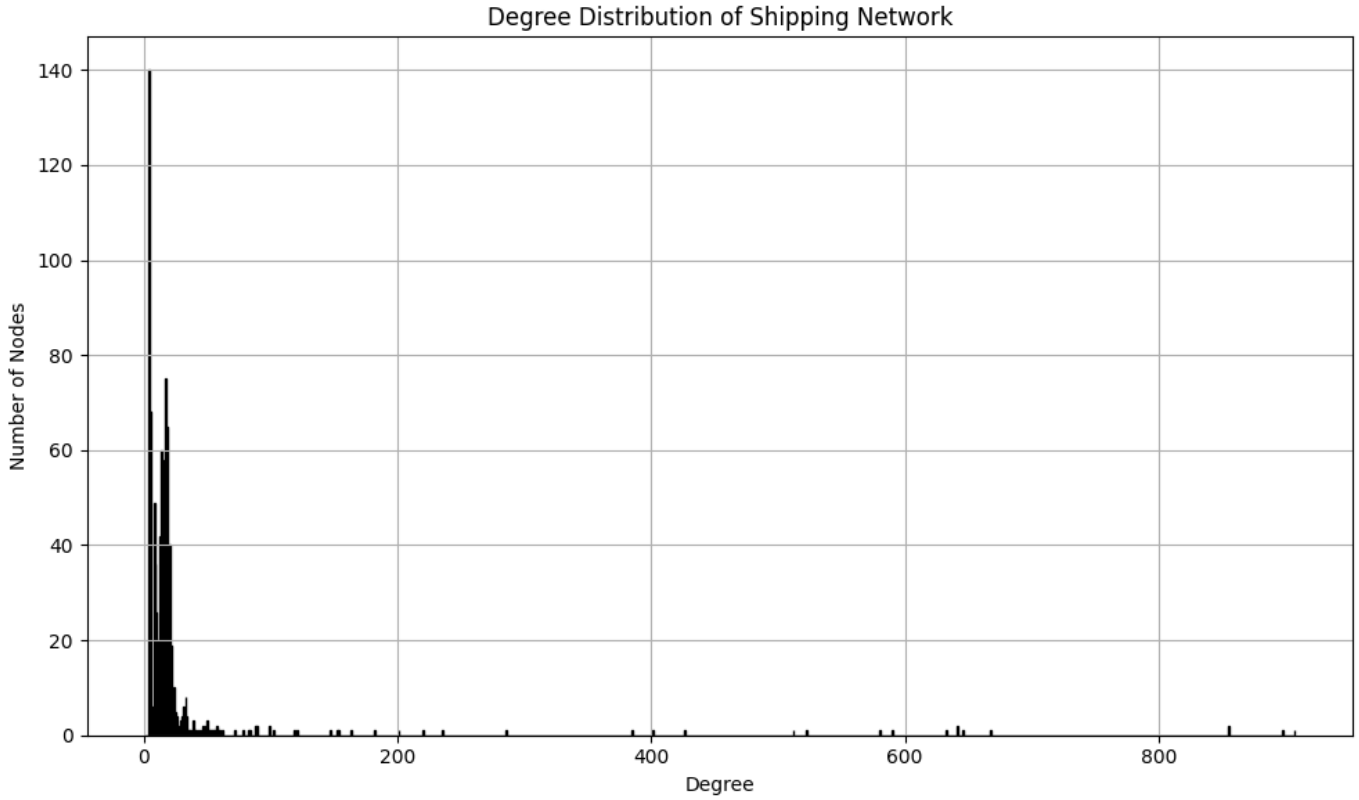
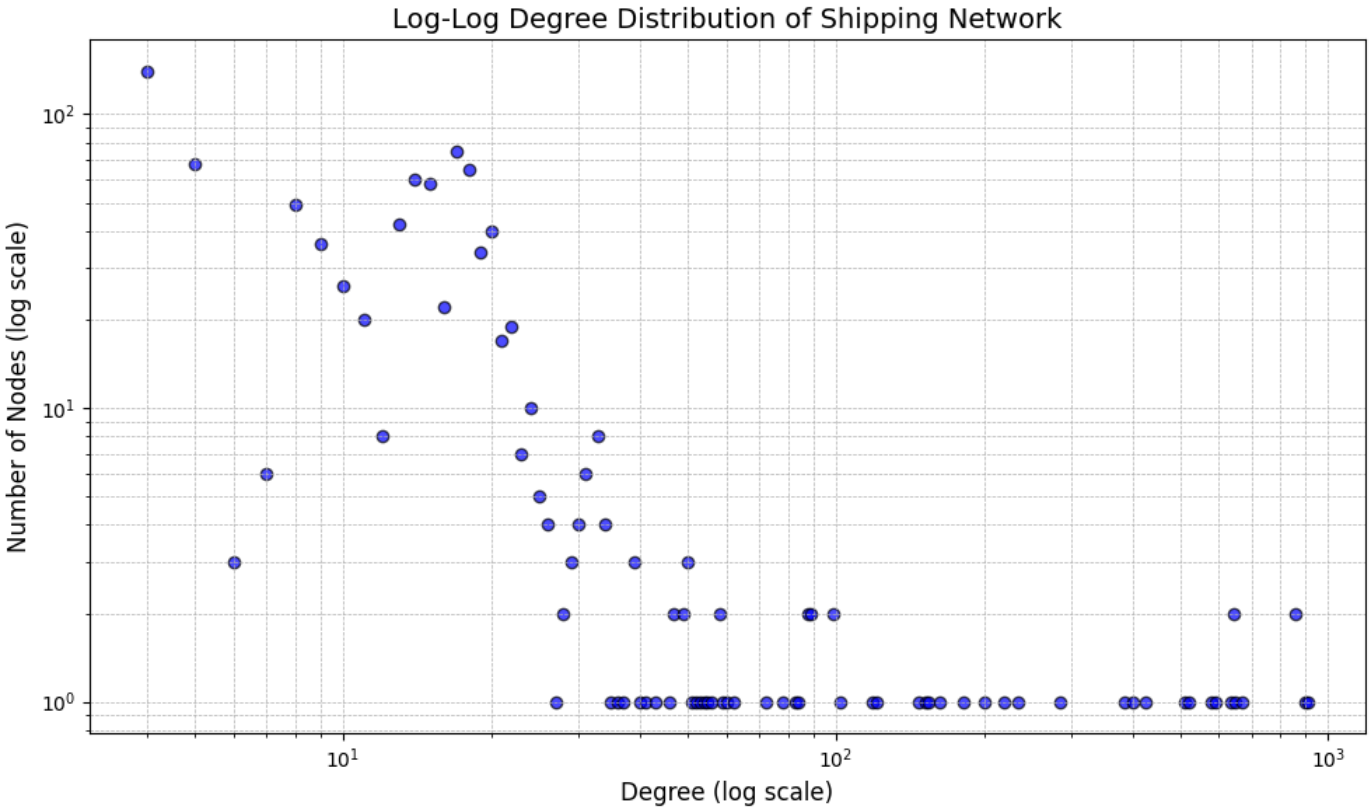
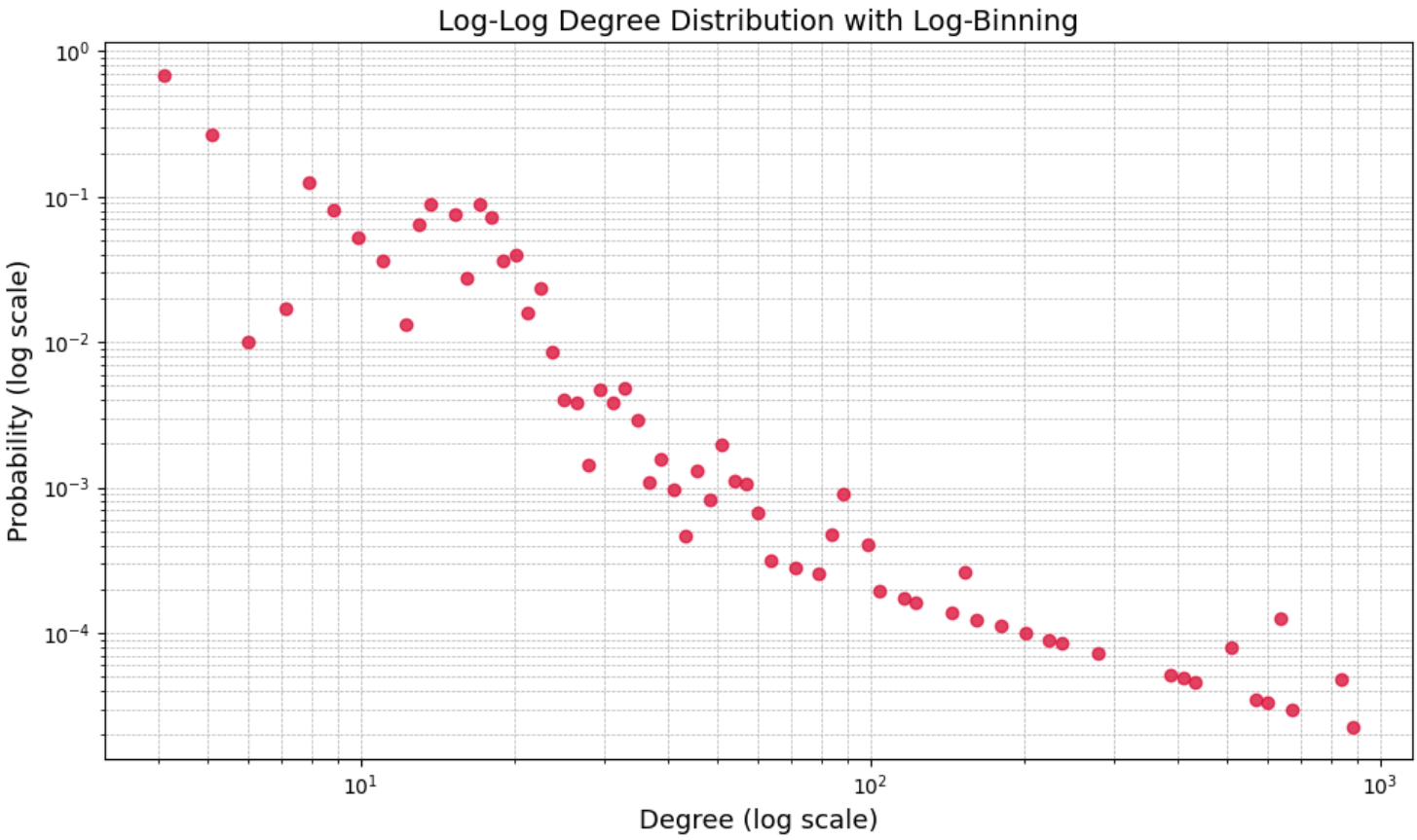



*Figure 1: There are small spikes for higher degree , this indicates presence of hubs*


Log-Log Degree Distribution of Shipping Network
Number of Nodes (log scale)
Degree (log scale)
$10^0$
$10^1$
$10^2$
$10^3$


Log-Log Degree Distribution with Log-Binning
Probability (log scale)
Degree (log scale)
$10^0$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^1$
$10^2$
$10^3$

**Notable Patterns:**

Average Degree**: 28.08 ,** Minimum Degree**: 4 ,** Maximum Degree: **907**

**1. Skewness Toward Low Degrees (Right Skewed):**

- Most nodes in the network have **low degrees**, while only a **few nodes have very high degrees**.
- This is clear in both the linear and log-log plots, with a sharp drop-off in the frequency of nodes as the degree increases.

**2. Heavy Tail / Scale-Free Behavior:**

- The log-log plot shows a **very heavy-tailed distribution**, where the tail approximately follows a straight line — a signature of **power-law or scale-free** behavior.
- This suggests the presence of **hubs** — a small fraction of of ports (nodes) that are highly connected, but this fraction is more as compared to normal power law suggesting good number of hubs , while the majority have very few connections.

**3. Power-Law Region (Linear in Log-Log):**

- There is a portion of the log-log plot that appears **approximately linear**, typically from mid-range degrees onwards.

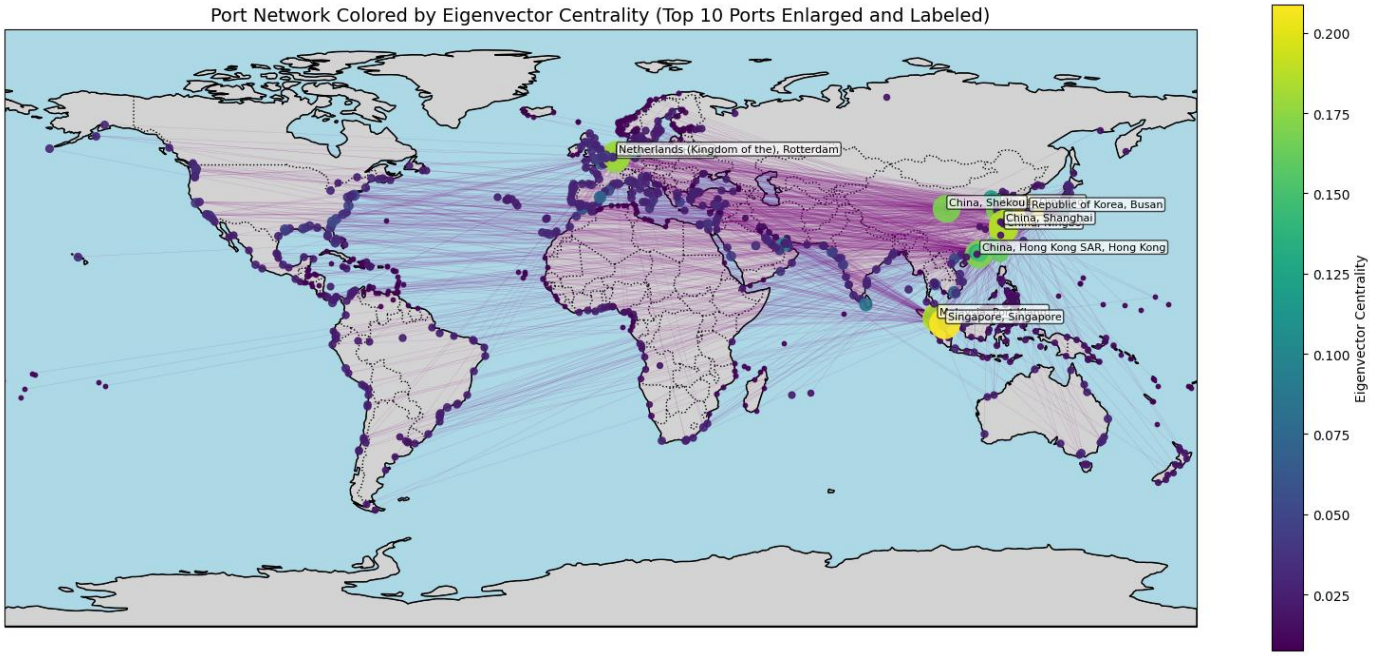
**Estimated power-law exponent (gamma): 1.8395** using MLE, this suggests that a gamma value (γ) less than 2 in a power-law degree distribution implies a dense, potentially very skewed network with a high probability of hubs (nodes with many connections) and a large mean degree that diverges.

Code to compute gamma using MLE :

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

# Load the GraphML file
G = nx.read_graphml(r'B:\Final Network Science Project\Data\grpah
file\Shipping_Network.graphml')

# Degree list (excluding 0-degree nodes)
degrees = np.array([d for n, d in G.degree()])
degrees = degrees[degrees > 0]

# Set minimum degree (k_min) for power-law fit
k_min = np.min(degrees)
# MLE formula to estimate gamma
n = len(degrees)
gamma = 1 + n / np.sum(np.log(degrees / k_min))
print(f"Estimated power-law exponent (gamma): {gamma:.4f}")
```

# Task 2: Centrality Analysis : code file: link

- **Eigenvector Centrality**
  **(**note; I have changed the exact location of Shekou, china port because it was overlapping with shanghai port**)**



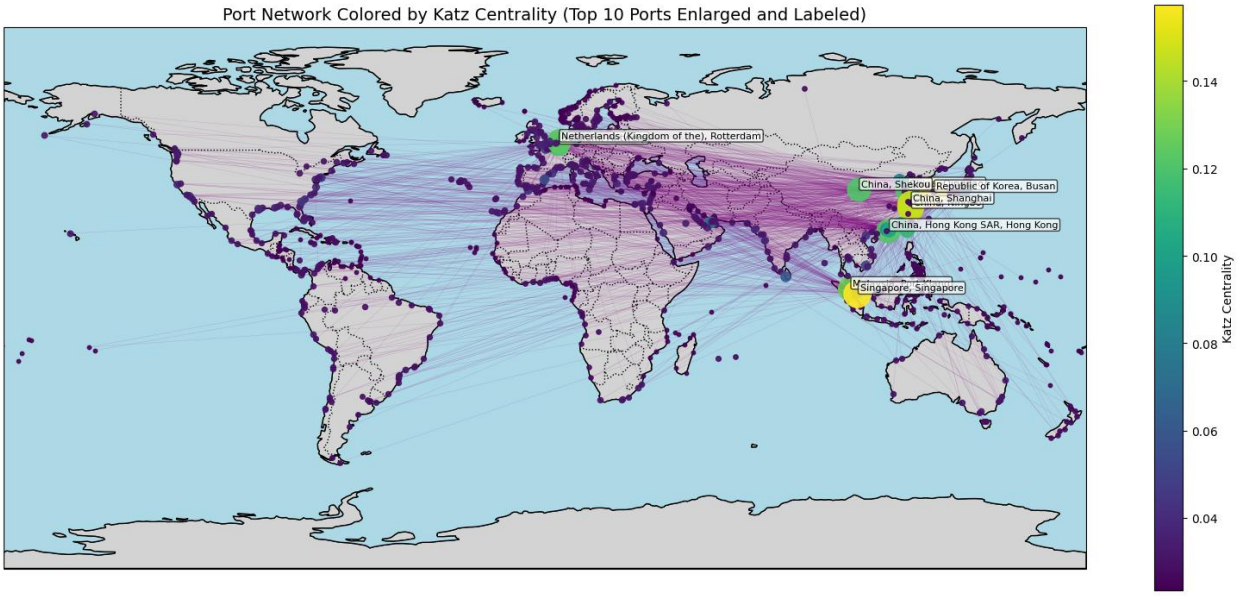

Top 10 nodes with highest Eigenvector Centrality :

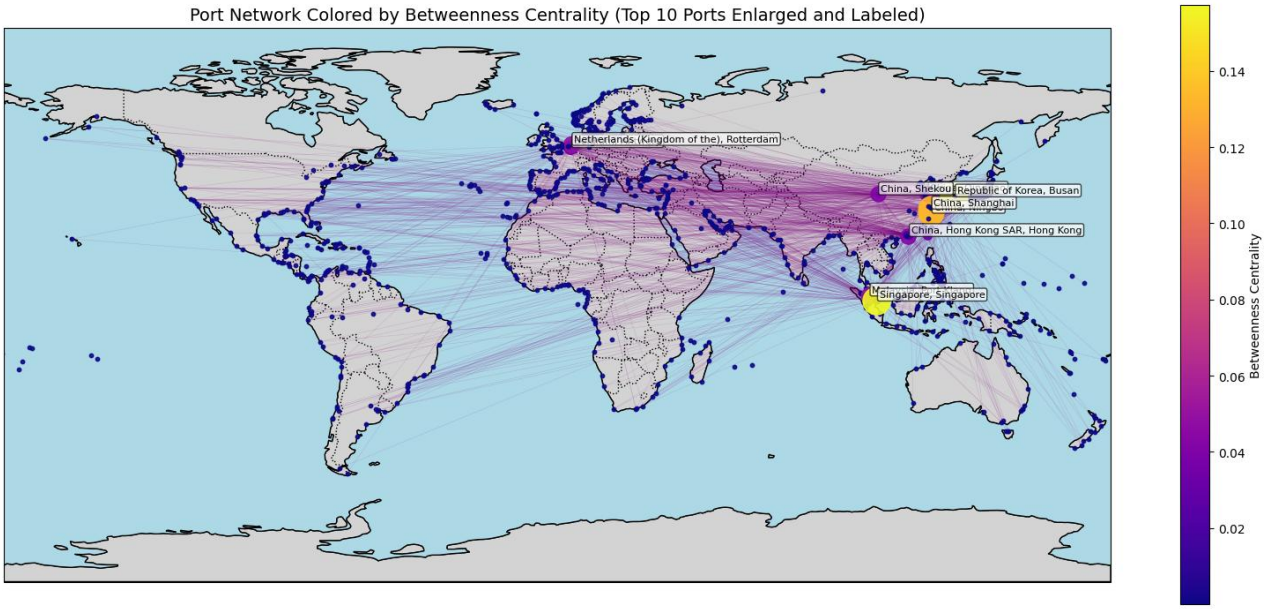
| | Port | Eigenvector Centrality |
|---|---|---|
| 0 | Singapore, Singapore | 0.208836 |
| 1 | Republic of Korea, Busan | 0.205705 |
| 2 | China, Ningbo | 0.186291 |
| 3 | China, Shanghai | 0.186291 |
| 4 | Malaysia, Port Klang | 0.185983 |
| 5 | Netherlands (Kingdom of the), Rotterdam | 0.178984 |
| 6 | Belgium, Antwerp | 0.175912 |
| 7 | China, Hong Kong SAR, Hong Kong | 0.170056 |
| 8 | China, Qingdao | 0.169503 |
| 9 | China, Shekou | 0.169503 |

- **Katz Centrality:**

Top 10 nodes:

| | Port | Katz Centrality |
|---|---|---|
| 0 | Singapore, Singapore | 0.157470 |
| 1 | Republic of Korea, Busan | 0.155899 |
| 2 | China, Ningbo | 0.146184 |
| 3 | China, Shanghai | 0.146184 |
| 4 | Malaysia, Port Klang | 0.128494 |
| 5 | Netherlands (Kingdom of the), Rotterdam | 0.123589 |
| 6 | China, Hong Kong SAR, Hong Kong | 0.121516 |
| 7 | China, Qingdao | 0.120894 |
| 8 | China, Shekou | 0.120894 |
| 9 | Belgium, Antwerp | 0.118586 |



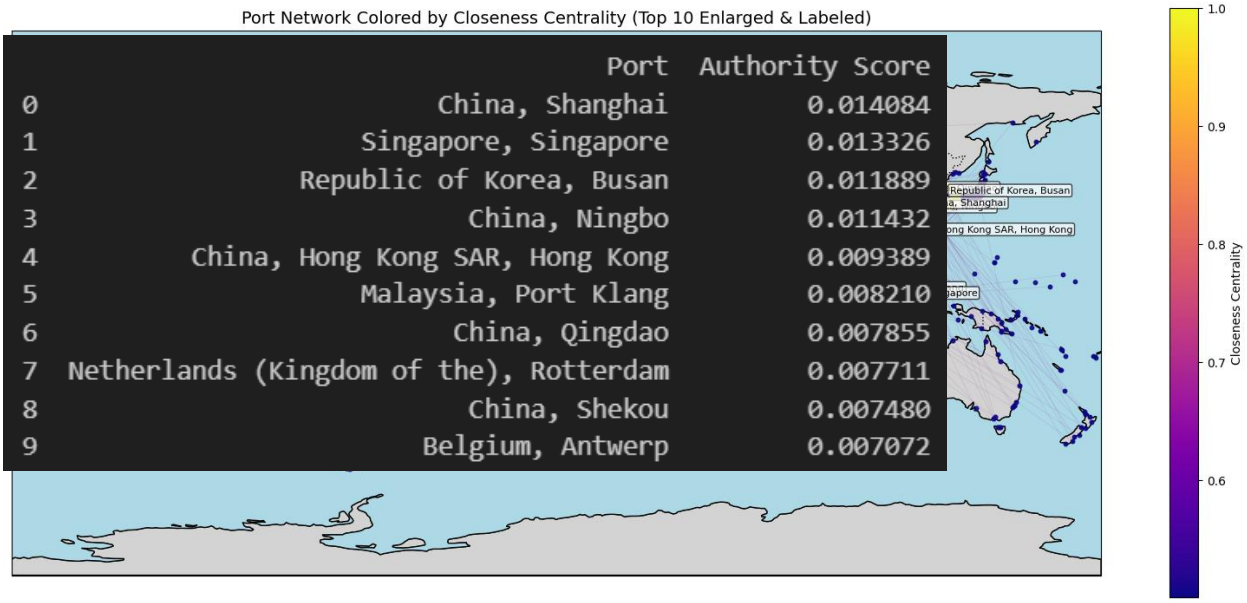



| | Port | Betweenness Centrality |
|---|---|---|
| 0 | Singapore, Singapore | 0.157315 |
| 1 | Republic of Korea, Busan | 0.154789 |
| 2 | China, Ningbo | 0.133193 |
| 3 | China, Shanghai | 0.133193 |
| 4 | Malaysia, Port Klang | 0.050066 |
| 5 | Netherlands (Kingdom of the), Rotterdam | 0.048422 |
| 6 | China, Hong Kong SAR, Hong Kong | 0.042519 |
| 7 | China, Qingdao | 0.041227 |
| 8 | China, Shekou | 0.041227 |
| 9 | Belgium, Antwerp | 0.033583 |

- **Betweenness Centrality**

- **Closeness Centrality**

Port Network Colored by Closeness Centrality (Top 10 Enlarged & Labeled)

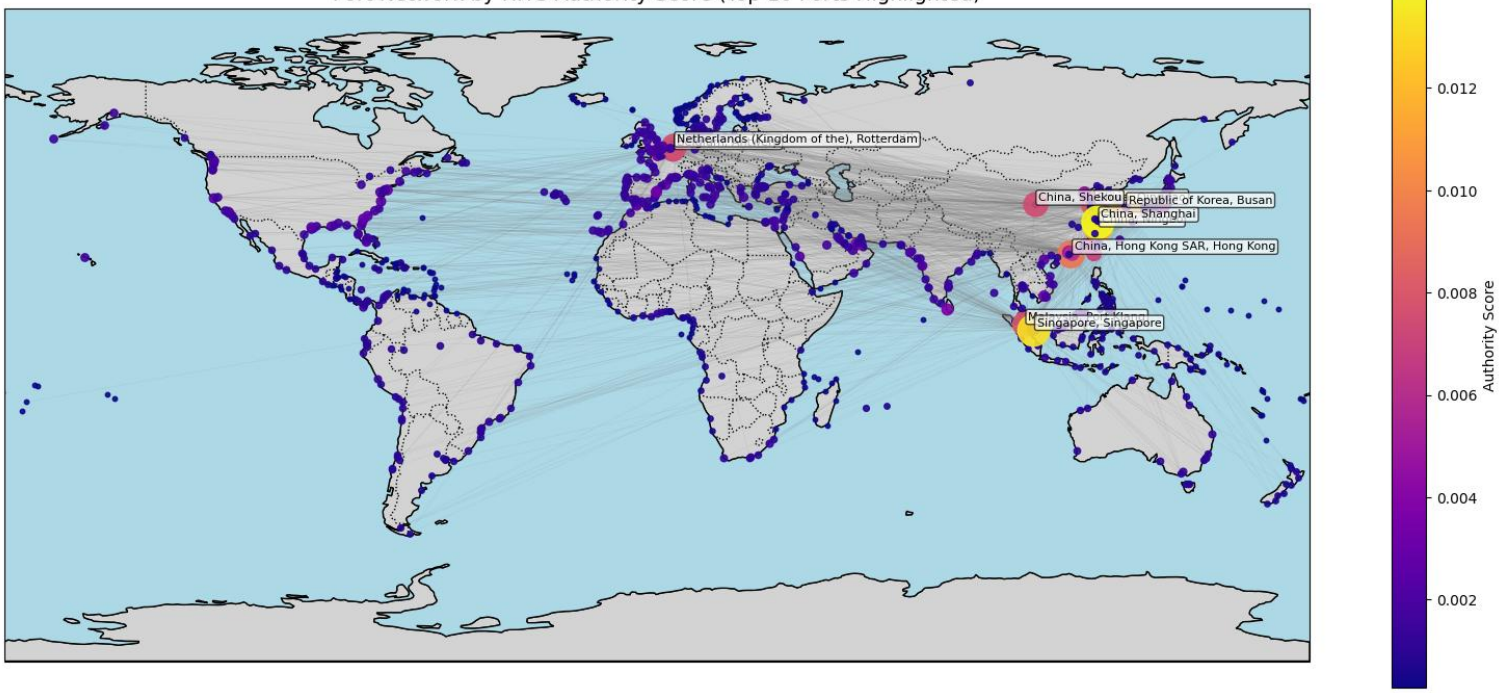



| | Port | Authority Score |
|---|---|---|
| 0 | China, Shanghai | 0.014084 |
| 1 | Singapore, Singapore | 0.013326 |
| 2 | Republic of Korea, Busan | 0.011889 |
| 3 | China, Ningbo | 0.011432 |
| 4 | China, Hong Kong SAR, Hong Kong | 0.009389 |
| 5 | Malaysia, Port Klang | 0.008210 |
| 6 | China, Qingdao | 0.007855 |
| 7 | Netherlands (Kingdom of the), Rotterdam | 0.007711 |
| 8 | China, Shekou | 0.007480 |
| 9 | Belgium, Antwerp | 0.007072 |

| | Port | Closeness Centrality |
|---|---|---|
| 0 | Singapore, Singapore | 1.000000 |
| 1 | Republic of Korea, Busan | 0.990175 |
| 2 | China, Ningbo | 0.945777 |
| 3 | China, Shanghai | 0.945777 |
| 4 | Malaysia, Port Klang | 0.791449 |
| 5 | China, Hong Kong SAR, Hong Kong | 0.776541 |

- **HITS (Authority and Hub Scores)**

Port Network by HITS Authority Score (Top 10 Ports Highlighted)



Why These Ports Are Central

The high centrality of these ports can be attributed to several factors:

1. **Strategic Geographic Location**: Singapore sits at the crucial Strait of Malacca connecting the Indian and Pacific Oceans, making it a natural transshipment point between East Asia and Europe. This explains its top ranking in three of four metrics.

2. **Infrastructure and Capacity:** Ports like Shanghai, Singapore, and Busan have invested heavily in port infrastructure, allowing them to handle massive cargo volumes and attract more shipping lines.

3. **Economic Importance:** Most top-ranked ports serve rapidly growing economies in Asia, particularly China. Shanghai and Ningbo handle enormous export volumes from China's manufacturing centers.

4. **Hub-and-Spoke System**: The global shipping network operates on a hub-and-spoke model where major ports serve as collection and distribution points. The ports with high centrality scores function as these critical hubs.

Comparison Across Centrality Measures

When comparing the rankings across different centrality measures:

1. Consistent Leaders: Singapore, Busan, and Chinese ports (Shanghai and Ningbo) consistently appear at the top across all measures, confirming their fundamental importance regardless of how centrality is defined.

2. Regional Patterns: Asian ports dominate all centrality rankings, reflecting Asia's growing role in global trade. European ports (Rotterdam and Antwerp) maintain significant positions but generally rank lower.

3. **Notable Shifts:**

    - **Shanghai ranks first in Authority score but third/fourth** in other measures
    - Rotterdam and Antwerp rank higher in Eigenvector centrality (5th and 6th) than in Closeness centrality (8th and 9th)
    - Belgium's Antwerp drops from 6th in Eigenvector to 9th in Authority and Katz measures

4. **Measure-Specific Insights:**

    - The perfect Closeness centrality score for Singapore (1.0) highlights its unique position as an ideal transshipment hub.
    - Shanghai's top Authority score suggests it's particularly important as a destination for shipping routes.

# 3: Modularity and Community Detection

**Code file link :** link , **Community CSV file**: link

- Use a community detection algorithm (e.g., Louvain or Girvan–Newman).
  I have Used Louvain community detection algorithm
- Compute the **modularity score**.

1. Identify how many communities exist in your network and present the community partition.

Ans-

```
import networkx as nx
import community as community_louvain
import pandas as pd

# Load graph
graphml_path = r'B:\Final Network Science Project\Data\grpah
file\Shipping_Network.graphml'
G = nx.read_graphml(graphml_path)

# Louvain community detection
partition = community_louvain.best_partition(G, weight='weight')

# Group nodes by community
from collections import defaultdict

community_dict = defaultdict(list)
for node, comm_id in partition.items():
    community_dict[comm_id].append(node)

# Log total number of communities
print(f"Total communities detected: {len(community_dict)}")
print("Communities and their sizes:")
for k, v in community_dict.items():
    print(f"Community {k}: {len(v)} nodes")

# Pad communities to the same length
max_len = max(len(nodes) for nodes in community_dict.values())
for k in community_dict:
    community_dict[k] += [None] * (max_len - len(community_dict[k]))

# Convert to DataFrame with each community as a column
```

```
community_df = pd.DataFrame({
    f"Community {k}": v for k, v in sorted(community_dict.items())
})

# Save to CSV
community_df.to_csv("louvain_communities_all.csv", index=False)
print("Saved complete community listing to 'louvain_communities_all.csv'")
```

```
Total communities detected: 3
Communities and their sizes:
Community 2: 455 nodes
Community 1: 329 nodes
Community 0: 124 nodes
```

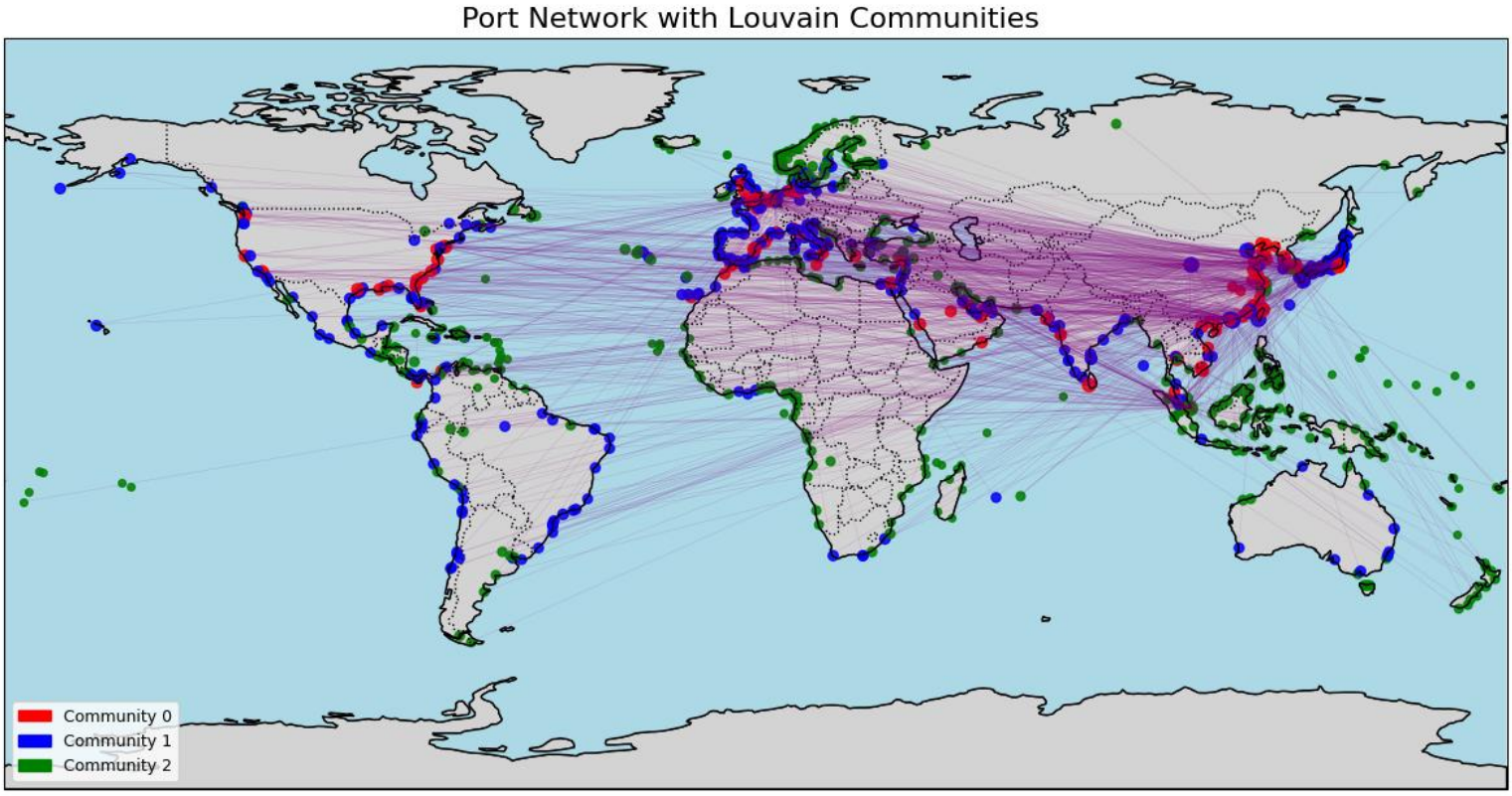
2. Report and interpret the modularity score. Does your network appear to have strong or weak community structure?

```
# Compute modularity score
modularity_score = community_louvain.modularity(partition, G, weight='weight')
print(f"Modularity Score: {modularity_score:.4f}")

# Interpretation
if modularity_score > 0.4:
    print("The network has a strong community structure.")
elif modularity_score > 0.3:
    print("The network has a moderately strong community structure.")
else:
    print("The network has a weak community structure.")

```

**Modularity Score: 0.1507**

The network has a weak community structure, because the modularity is less than 0.3 and there are only three communities detected.

3. visualizing the network with distinct colors for different communities.




4. Provide insights or interpretations: Are the communities meaningful in the context of your data?

**Community 0 (Red)**
The red community appears to represent the primary global shipping backbone, including major hub ports that you've correctly identified (Shanghai, Hong Kong, Singapore, Korean ports, Sri Lanka, Mundra, UAE, San Francisco, and New York). These ports share several important characteristics:

- They form what shipping experts call the "main trunk routes" - the critical east-west shipping lanes that handle the highest volume of global trade.
- They're predominantly located along the busy Asia-Europe-North America triangle that dominates global container shipping.
- The dense connections (purple lines) between these ports indicate they function as transshipment hubs where cargo is consolidated and redistributed.

**Community 1 (Blue)**
The blue community forms a secondary network layer with different characteristics:

- These ports appear more distributed globally, particularly along coastlines of South America, Western Africa, parts of Europe, and Australia.

- They likely function as regional connectors that feed into and receive cargo from the major red community hubs.
- Their position in the network suggests they handle significant regional trade but may not have the same level of intercontinental connectivity as red ports.

**Community 2 (Green)**
The green community consists of more peripheral ports:

- These are concentrated in northern Europe, parts of Southeast Asia, and scattered in other regions.
- They likely represent more specialized ports (perhaps bulk cargo, oil terminals) or those serving primarily local markets.
- Their limited connections to the main network suggest they may be either ports serving niche markets, or feeder ports for nearby larger hubs.

## Task 4: Assortativity & Degree-Degree Correlations

Code: link

- **Degree Assortativity:** Do high-degree nodes connect preferentially to other high-degree nodes?

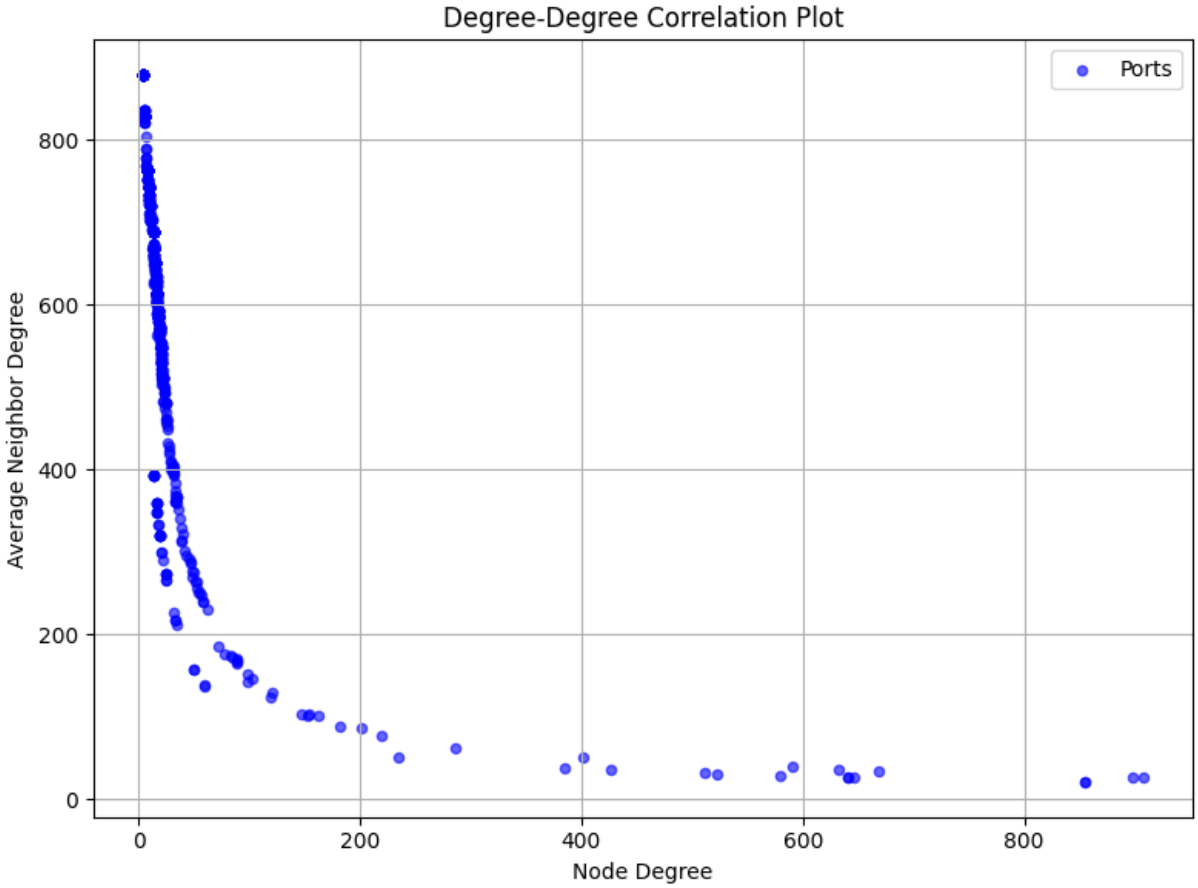
Ans- Pearson Correlation coefficient:

```
import networkx as nx

# Load your graph
graphml_file = r"B:\Final Network Science Project\Data\grpah
file\Shipping_Network.graphml"
G = nx.read_graphml(graphml_file)

# Calculate degree assortativity coefficient (Pearson correlation)
assortativity_coeff = nx.degree_assortativity_coefficient(G)

print(f"Degree Assortativity Coefficient (Pearson): {assortativity_coeff:.4f}")
```

Degree Assortativity Coefficient (Pearson): **-0.6889**

- **Degree-Degree Correlation Plot:** Plot average neighbor degree vs. node degree.

```
import matplotlib.pyplot as plt
import numpy as np

# Compute degree and average neighbor degree for all nodes
degrees = dict(G.degree())
avg_neighbor_degrees = nx.average_neighbor_degree(G)

# Prepare data for plotting
degree_list = []
avg_neighbor_degree_list = []

for node in G.nodes():
    k = degrees[node]
    knn = avg_neighbor_degrees[node]
    degree_list.append(k)
    avg_neighbor_degree_list.append(knn)

# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(degree_list, avg_neighbor_degree_list, alpha=0.6, s=20, color='blue',
label='Ports')



plt.xlabel("Node Degree")
plt.ylabel("Average Neighbor Degree")
plt.title("Degree-Degree Correlation Plot")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```




The network has a **degree assortativity coefficient of -0.6889**, which indicates a **strongly negative correlation**.

This means the network is **disassortative**, where:

- **High-degree nodes** (such as major shipping ports) tend to connect with **low-degree nodes** (smaller or regional ports).
- This suggests a **hub-and-spoke** structure, which is common in transportation and shipping networks.

**Relevance to Shipping Networks**

In global shipping systems:

- Large central hubs (like Shanghai, Singapore, Rotterdam) are connected to many smaller ports.
- This structure helps optimize efficiency by centralizing cargo handling and reducing direct connections.
- However, it also implies that the network could be more vulnerable to disruptions at these high-degree hubs.

## Task 5: Clustering Coefficients: code : link

- Compute the **global clustering coefficient** (transitivity).

```
import networkx as nx

# Load the graph
graphml_path = r"B:\Final Network Science Project\Data\grpah
file\Shipping_Network.graphml"
G = nx.read_graphml(graphml_path)

# Global clustering coefficient
global_clustering = nx.transitivity(G)
print(f"Global Clustering Coefficient (Transitivity): {global_clustering:.4f}")
```

Global Clustering Coefficient (Transitivity): **0.0685**

Generating a Random graph of similar size and density:

```
# Parameters of original graph
n = G.number_of_nodes()
m = G.number_of_edges()

# Generate Erdős–Rényi random graph with same nodes and edge probability
p = (2 * m) / (n * (n - 1))
G_random = nx.erdos_renyi_graph(n, p)

# Global clustering for random graph
random_clustering = nx.transitivity(G_random)
print(f"Random Graph Clustering Coefficient: {random_clustering:.4f}")
```

Random Graph Clustering Coefficient: **0.0302**

**Higher Clustering in Original Graph:**
The clustering coefficient of your real-world shipping network (0.0685) is more than double that of a random graph with the same number of nodes and edges (0.0302).

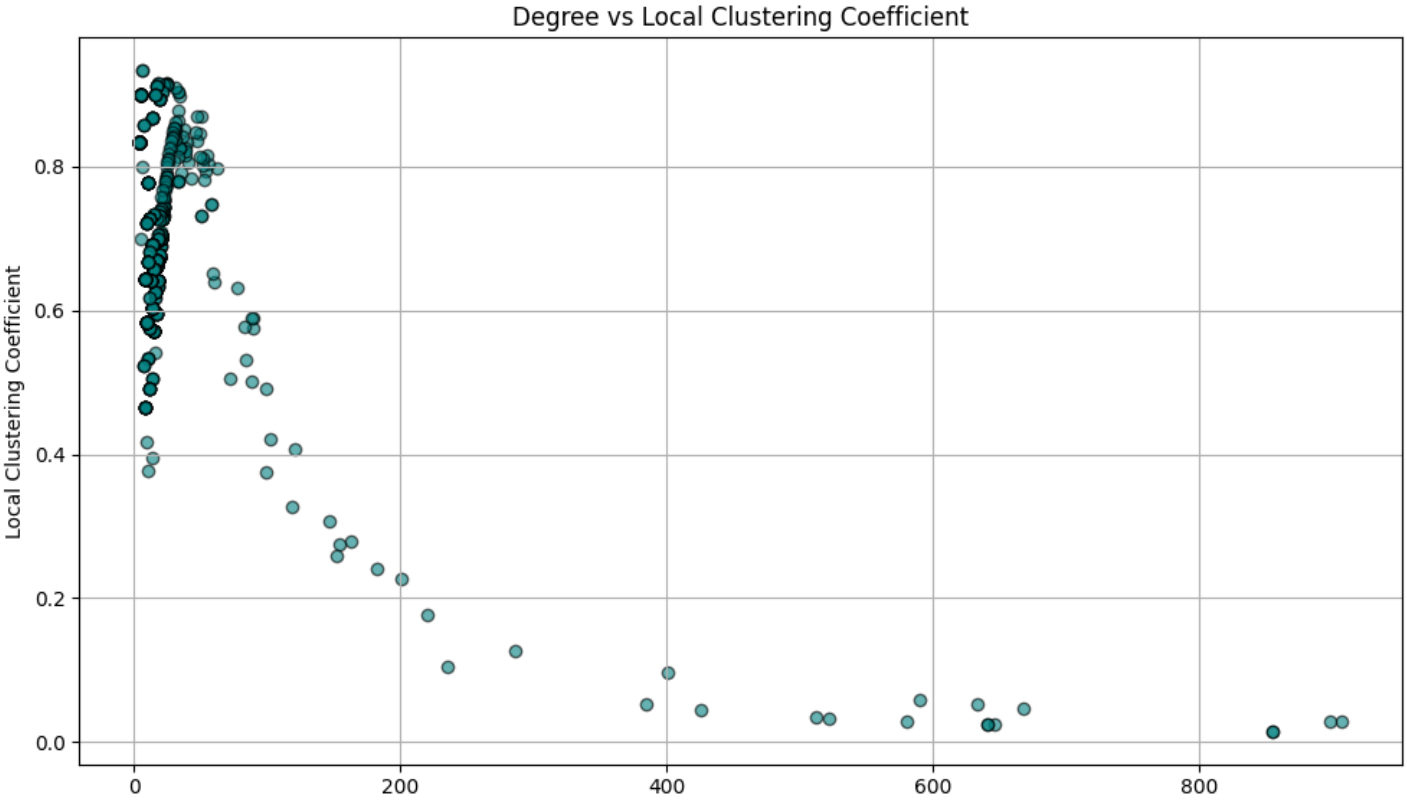
**Implication:**
This suggests that ports in your network tend to form tightly-knit clusters — if Port A is connected to Port B and Port C, there's a higher-than-random chance that B and C are also connected.

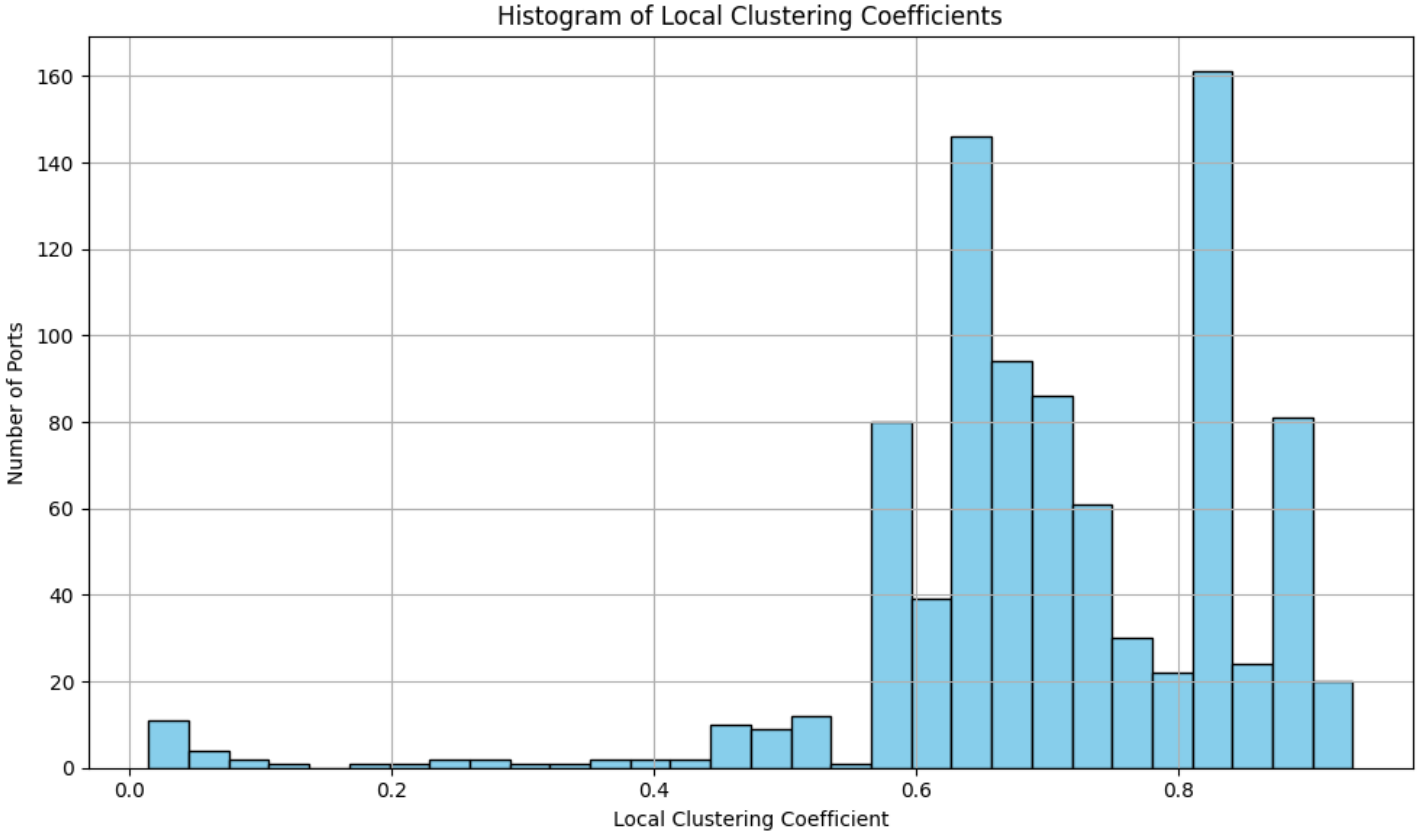
**Real-World Insight:**
In the context of a shipping network, this is likely to reflect regional shipping hubs or alliances, where certain sets of ports have strong mutual connectivity — for example, due to trade routes, proximity, or logistical agreement.

- Compute local clustering coefficients for each node (or for a subset).
  Link to the Local clustering coefficient csv file: link

```
import networkx as nx
import pandas as pd

# Load the graph
graphml_path = r'B:\Final Network Science Project\Data\grpah
file\Shipping_Network.graphml'
G = nx.read_graphml(graphml_path)

# Compute local clustering coefficients
local_clustering = nx.clustering(G)
# Compute degree of each node
degree_dict = dict(G.degree()
# Combine into DataFrame
clustering_df = pd.DataFrame({
    'Port': list(G.nodes()),
    'Degree': [degree_dict[node] for node in G.nodes()],
    'Local Clustering Coefficient': [local_clustering[node] for node in G.nodes()]
})

clustering_df = clustering_df.sort_values(by='Degree', ascending=False)
clustering_df.to_csv("clustering_by_degree_full.csv", index=False)

print("data saved to 'clustering_by_degree_full.csv'")
```

| Port | Degree | Local Clustering Coefficient |
|---|---|---|
| Singapore, Singapore | 907 | 0.028821698 |
| Republic of Korea, Busan | 898 | 0.028451681 |
| China, Shanghai | 855 | 0.014999247 |
| China, Ningbo | 855 | 0.014999247 |
| Malaysia, Port Klang | 668 | 0.047410427 |
| China, Hong Kong SAR, Hong Kong | 646 | 0.023937409 |
| China, Qingdao | 641 | 0.024229719 |
| China, Shekou | 641 | 0.024229719 |
| Netherlands (Kingdom of the), Rotterdam | 633 | 0.05164277 |
| Belgium, Antwerp | 590 | 0.058732123 |


Degree vs Local Clustering Coefficient
Local Clustering Coefficient
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800


Histogram of Local Clustering Coefficients
Number of Ports
0
20
40
60
80
100
120
140
160
0.0
0.2
0.4
0.6
0.8
Local Clustering Coefficient