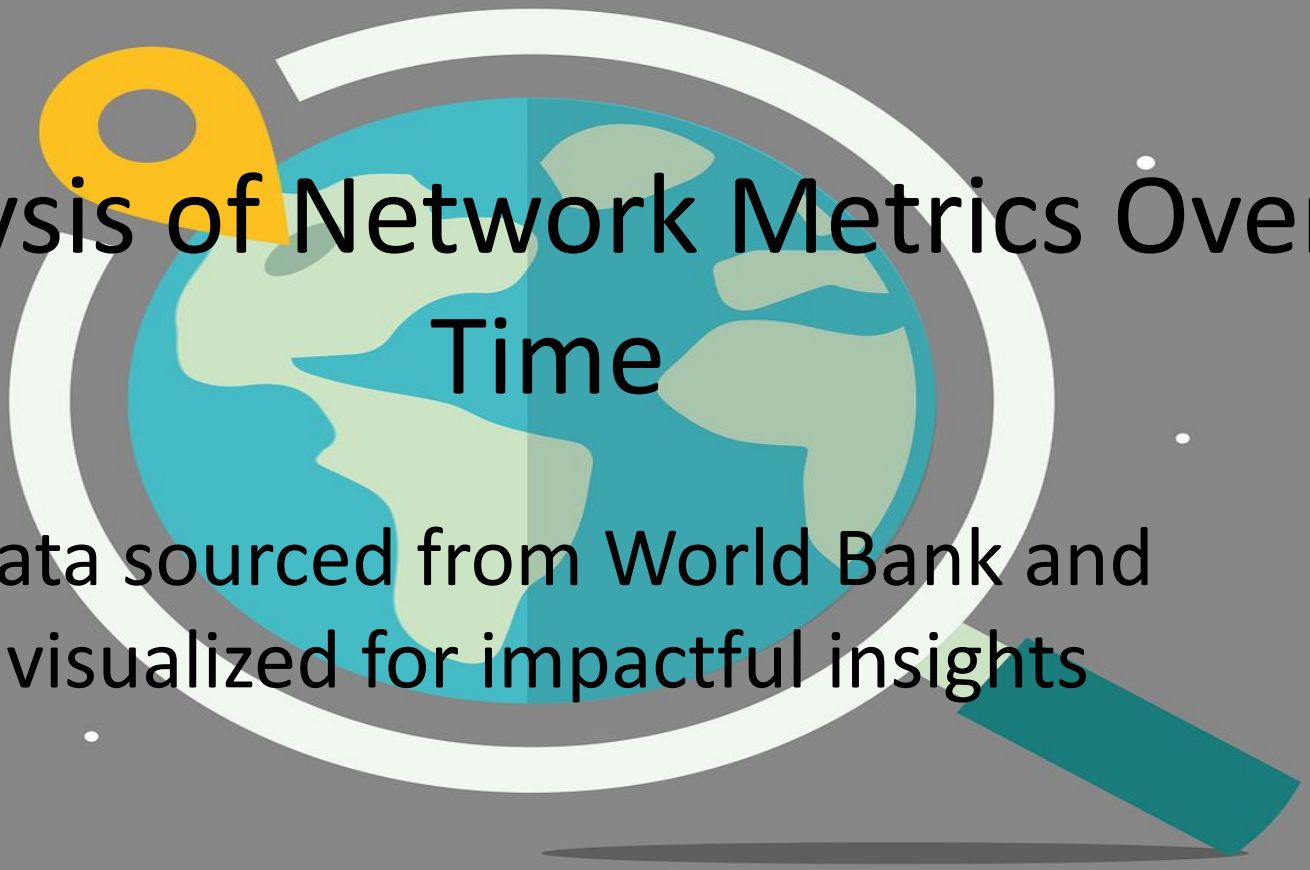
Analysis of Network Metrics Over Time

Data sourced from World Bank and visualized for impactful insights



Deliverables

1)Making a directed weighted network where we only accept an edge if it satisfies a certain threshold percentage of imports/exports w.r.t Total trade.

2) We analyse the dataset of trade between the countries over the years to find out the emerging clusters of trade networks between the countries over time.

3) We induce hypothetical sanctions and observe the changes in the network properties

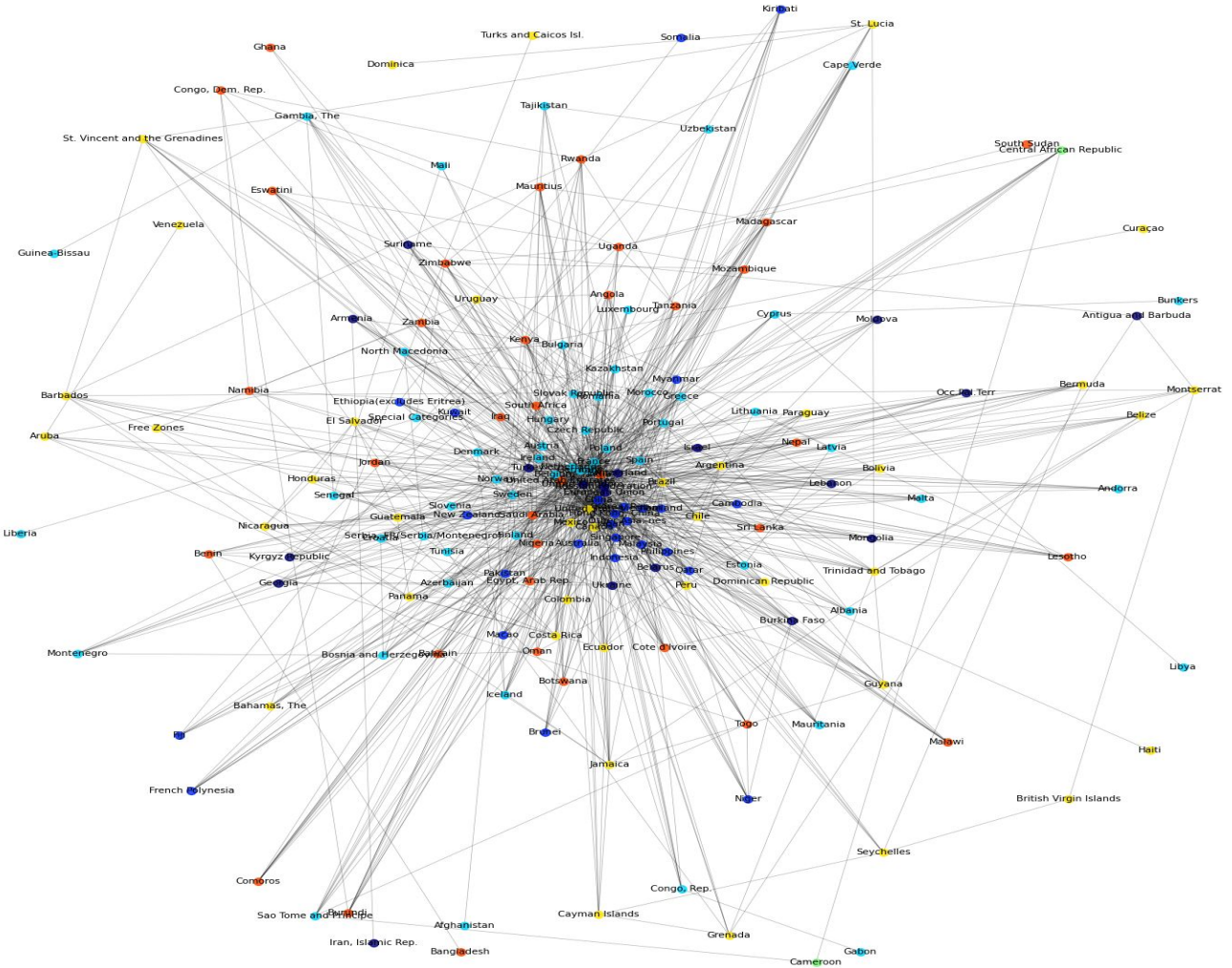
4)We observe the effect of a random node deletion on the network by analysing the respective change in clustering coefficient.

5)We examine the effect of major world events covering the China's accession to WTO (2001), financial crisis of 2008 and the Brexit (2016).

For our project, We intend to use the data available from WITS Trade Set database(https://wits.worldbank.org/countrystats.aspx?lang=en). The nodes of our network would be respective countries and edges would be the import and exports making the graph weighted and directed.

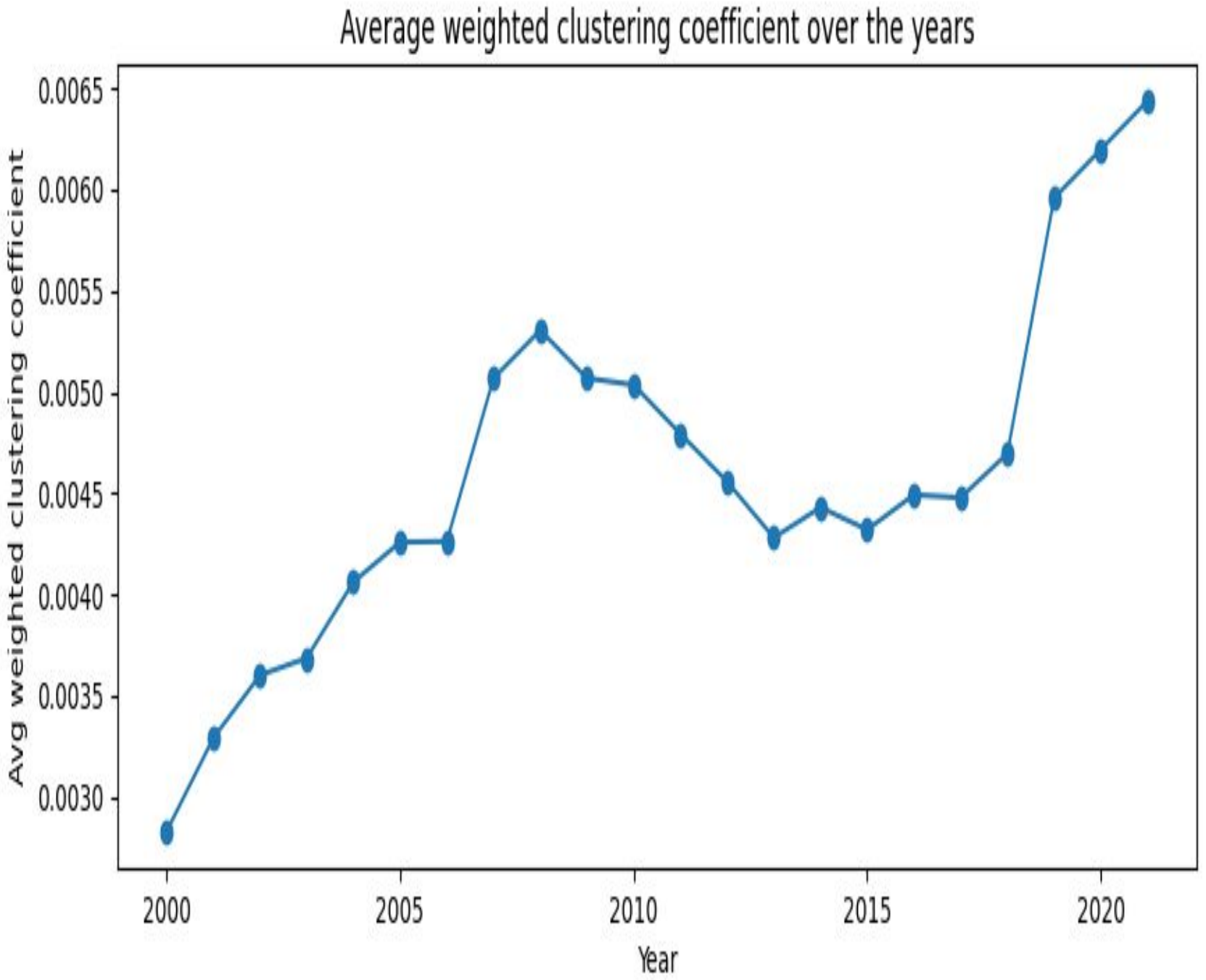
Functions to make directed weighted graph

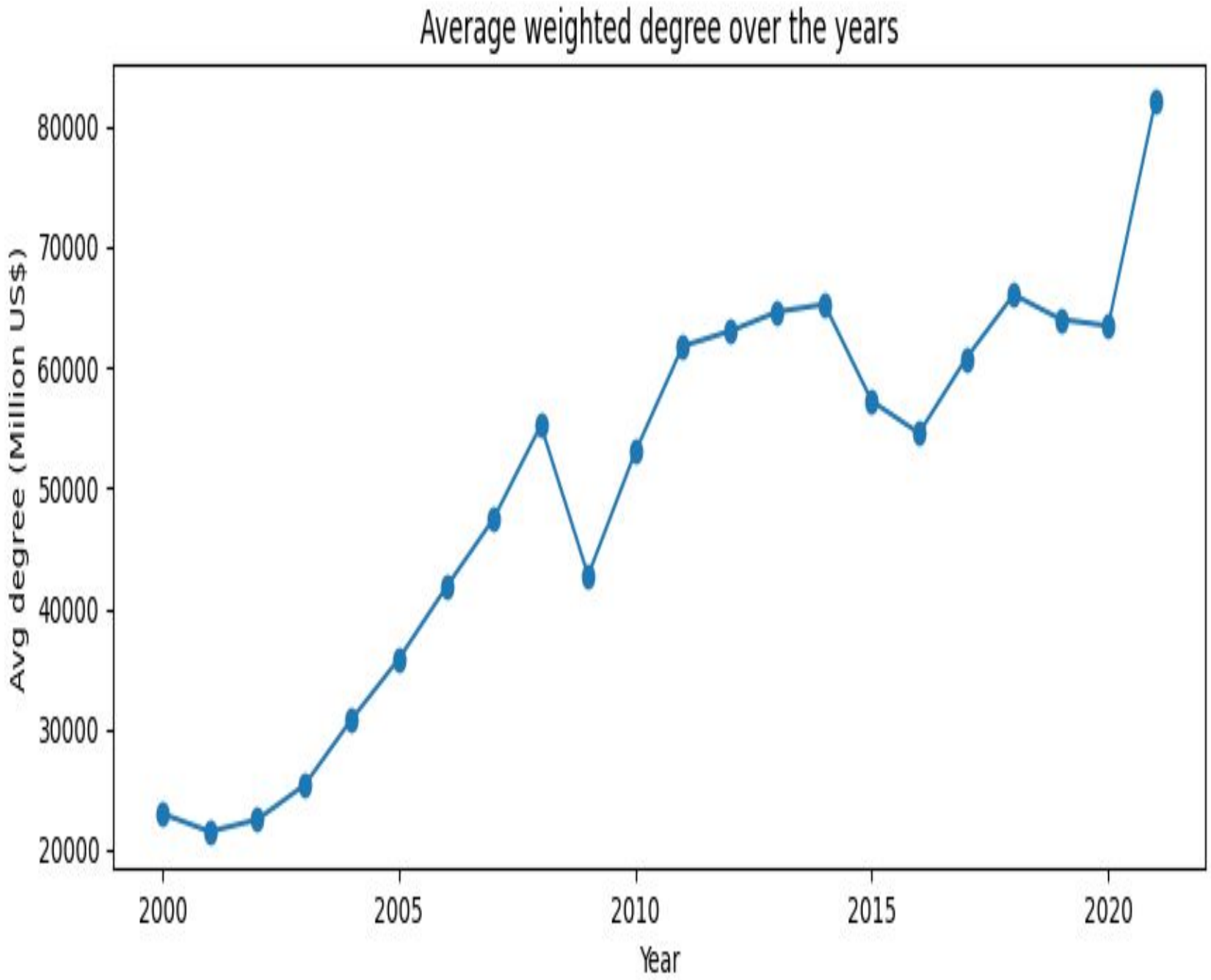
```
def add_country(year):
    df=pd.DataFrame({'Reporter':[],'Partner':[],'Indicator Type': [],'Indicator':[],str(year):[]})

    for filename in os.listdir("wits_en_trade_summary_allcountries_allyears"):
        df1=pd.read_csv(f"wits_en_trade_summary_allcountries_allyears/{filename}",encoding='latin-1')

        df1_1=df1[['Reporter','Partner','Indicator Type','Indicator',str(year)]].dropna()
        df1_1=df1_1[df1_1['Reporter']!='World']
        df1_1=df1_1[df1_1['Partner']!='World']
        df1_1=df1_1[df1_1['Partner']!=' World']
        df1_1=df1_1[df1_1['Partner']!='Unspecified']
        df1_1=df1_1[df1_1['Partner']!='...']
        df1_1=df1_1[df1_1['Indicator']!='Partner share(%)-Top 5 Import Partner']
        df1_1=df1_1[df1_1['Indicator']!='Partner share(%)-Top 5 Export Partner']

        df1_1=df1_1.reset_index()
        for index, rows in df1_1.iterrows():
            # print(index,rows)
            if rows['Indicator Type']=='Import':
                df1_1.iloc[index].Reporter, df1_1.iloc[index].Partner=rows['Partner'],rows['Reporter']
            # elif rows['Indicator Type']=='Export':
            #     df1_1.iloc[index].Reporter, df1_1.iloc[index].Partner=rows['Partner'],rows['Reporter']

        # print(df1_1)
        df=pd.concat([df,df1_1])
```

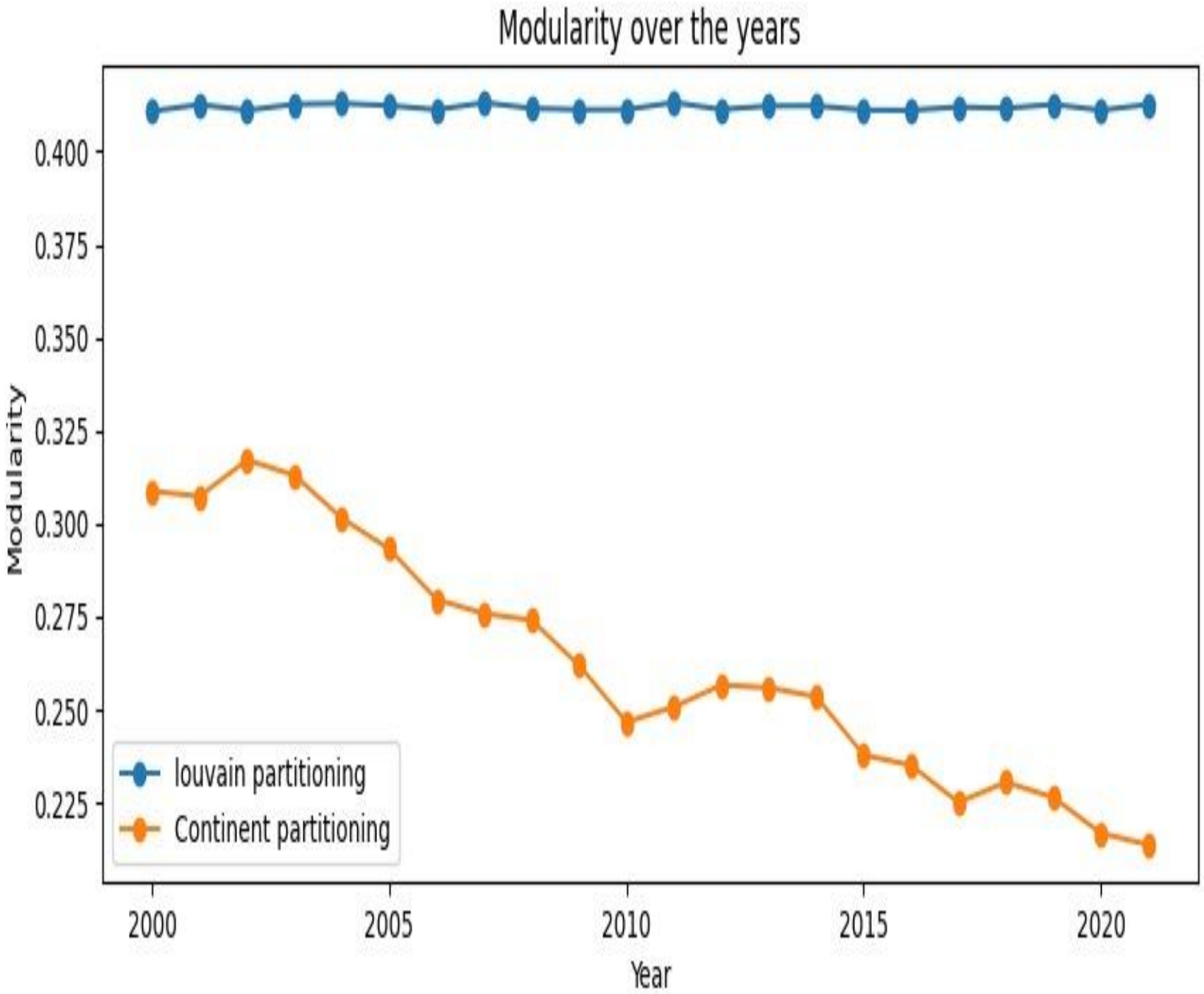


Graphs have been visually represented on an annual basis spanning from the year 1993 through 2021, ensuring a comprehensive and detailed visualization of data trends across this significant timeframe.

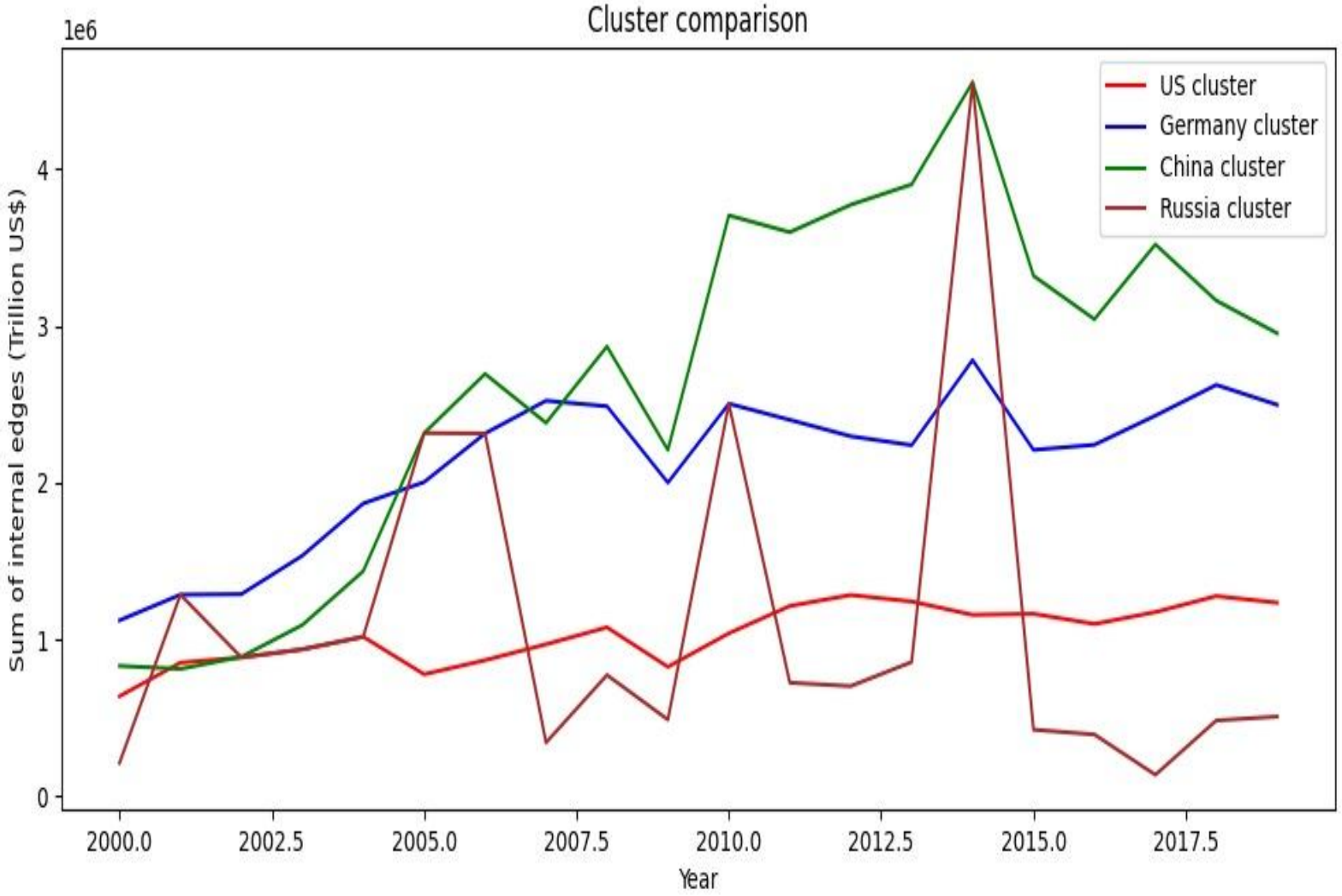
General trends of the network from 2000-2021


Average weighted clustering coefficient over the years
Avg weighted clustering coefficient
0.0065
0.0060
0.0055
0.0050
0.0045
0.0040
0.0035
0.0030
2000
2005
2010
2015
2020
Year


Average weighted degree over the years
Avg degree (Million US$)
80000
70000
60000
50000
40000
30000
20000
2000
2005
2010
2015
2020
Year


Modularity over the years
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
Modularity
louvain partitioning
Continent partitioning
2000
2005
2010
2015
2020
Year

Clusters over time



SANCTIONS

AIM: Simulating situations where country A imposes sanctions on country B.

1)When country applies sanction on another country, we want the neighbours i of country A in the network who share a weighted edge i.e the total trade with country A be over 5% of the total trade of neighbour i , they must also apply sanctions on country B iff weight(A,i)> weight(B,i).

2)This helps us propagate the effect of sanctions over the whole network. It also sits well with the fact that countries with little economic power aren't able to coerce other trade partners to also put sanctions while allowing major players to use economic influence.

3)The function also reports alternative countries for different industries. Based on export data obtained from the network, we can find which countries would benefit from the sanctions. Parameters for the year are also provided.Also, further optimisations and functionalities can be added to the function.

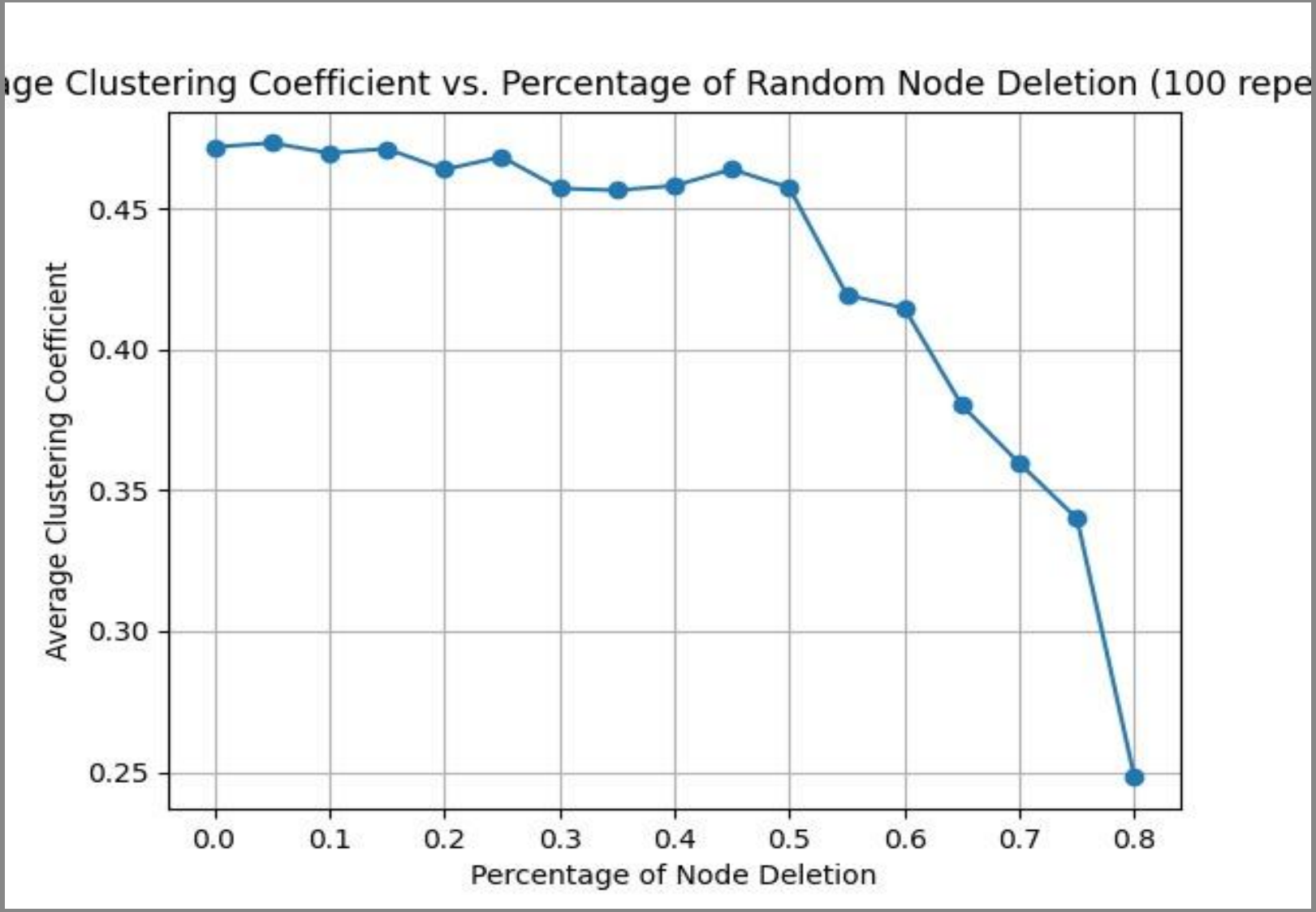
Sample input: sanction(‘India’, ‘United states,2017)

```
Countries affected directly: 22
Most valuable trading partners lost: ['China', 'United States', 'Indonesia', 'Singapore', 'Australia', 'South Africa', 'Ukraine', 'Peru', 'Nigeria', 'Oman']
Total trade deficit(US$ Million):  153428.67000000004
Total trade deficit (as % of trade lost): 46.361000098506274
Total trade deficit (as % of world trade):1.3148554088886482
                                  Before              After
 Degree_centrality   0.2513089005235602 0.07853403141361257
Closeness_centrality 0.53954802259887    0.3979166666666664
Betweenness      0.024881827936033822  0.0012478394624464684
Top alternatives for Consumer goods
China : 824787.78,European Union : 756862.73,Germany : 513675.08,United States : 400719.11,Italy : 218988.36,France : 198186.57,Netherlands : 186063.59,Japan : 172271.53,United Kingdom : 167224.52


...
```

```python
def find_exports(country,year):
    exports=[]
    for filename in os.listdir("wits_en_trade_summary_allcountries_allyears"):
        df1=pd.read_csv(f"wits_en_trade_summary_allcountries_allyears/{filename}",encoding='latin-1')
        if len(df1)==0 or df1.iloc[0]['Reporter']!=country:
            continue
        else:
            df1_1=df1[['Reporter','Partner','Product categories','Indicator Type','Indicator',str(year)]].dropna()

            df1_1=df1_1[df1_1['Partner']=='World']
            df1_1=df1_1[df1_1['Indicator Type']=='Export']
            df1_1=df1_1[df1_1['Indicator']=='Export(US$ Mil)']

            df1_1=df1_1[df1_1['Product categories']!='All products']
            # print(df1_1)
            for index,row in df1_1.iterrows():
                if float(row[str(year)])>5000:
                    exports.append((row['Product categories'],float(row[str(year)])))

            return exports
```

```
def sanction(country,country1,year):
    G=undirected(year)

    # print(list(G.neighbors(country)))
    # retain=[i for i in input("Enter Countries you want to retain:").split(",")]
    retain=[country]
    for n in G.neighbors(country1):
        if n!=country:
            degree = sum(weight for _, _, weight in G.edges(n, data='weight'))
            if (G[country1][n]['weight']/degree) * 100<5 or (G.has_edge(country,n) and G[country][n]['weight']>G[country1][n]['weight']) :
                retain.append(n)
    print(f"Countries affected directly: {len(list(G.neighbors(country)))-len(retain)}")
    print("\nMost valuable trading partners lost:",find_top(G,country,retain))
    degree_centrality = nx.degree_centrality(G)[country]
    closeness_centrality = nx.closeness_centrality(G)[country]
    betweenness_centrality = nx.betweenness_centrality(G)[country]
    exports=find_exports(country,year)
    Tot_trade1=sum([G[country][n]['weight'] for n in list(G.neighbors(country))])
    Tot_trade=sum([G[country][n]['weight'] for n in list(G.neighbors(country)) if n not in retain])
    print("\nTotal trade deficit(US$ Million): ",Tot_trade)
    total_weight = sum(weight for _, _, weight in G.edges(data='weight', default=1))
    print("\nTotal trade deficit (as % of trade lost): ",Tot_trade/Tot_trade1 * 100)
    print("\nTotal trade deficit (as % of world trade): ",Tot_trade/total_weight * 100)
    for n in list(G.neighbors(country)):
        if n not in retain:
            G.remove_edge(country,n)

    degree_centrality1 = nx.degree_centrality(G)[country]
    closeness_centrality1 = nx.closeness_centrality(G)[country]
    betweenness_centrality1 = nx.betweenness_centrality(G)[country]
    print(f"\n\t\t\t\tBefore\tAfter\nDegree_centrality {degree_centrality}\t{degree_centrality1}\nCloseness_centrality {closeness_centrality}\t
{closeness_centrality1}\nBetweenness_centrality {betweenness_centrality}\t {betweenness_centrality1}")

    for export in exports:
        print("\n")
        top_alts = sorted(All_prods[year][export[0]], key=lambda x: x[1], reverse=True)
        print(f"Top alternatives for {export[0]}")
        for i in top_alts[1:10]:
            if i[0]!=country:
                print(i[0],":",i[1])
```

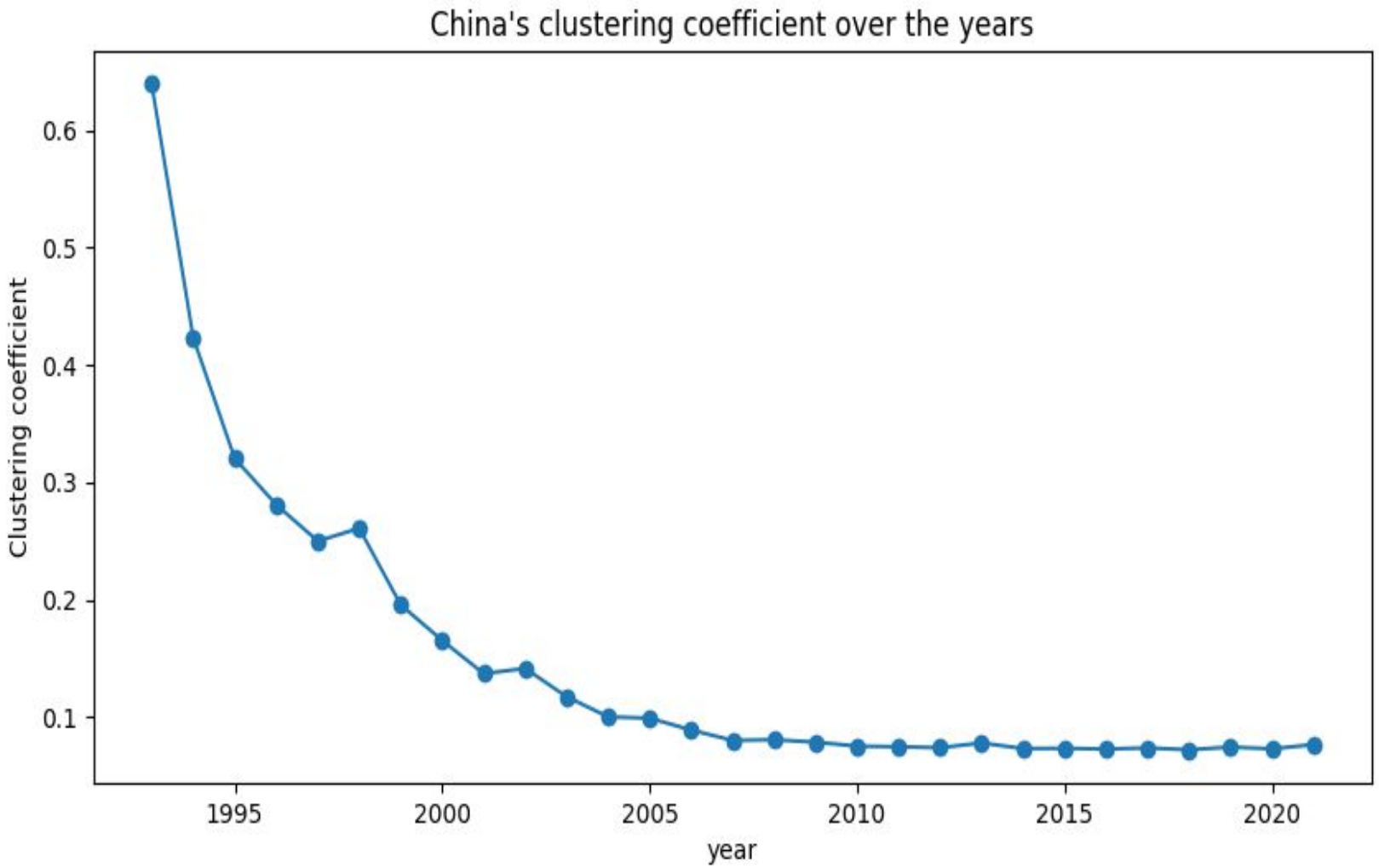
Clustering coefficient in random node deletion



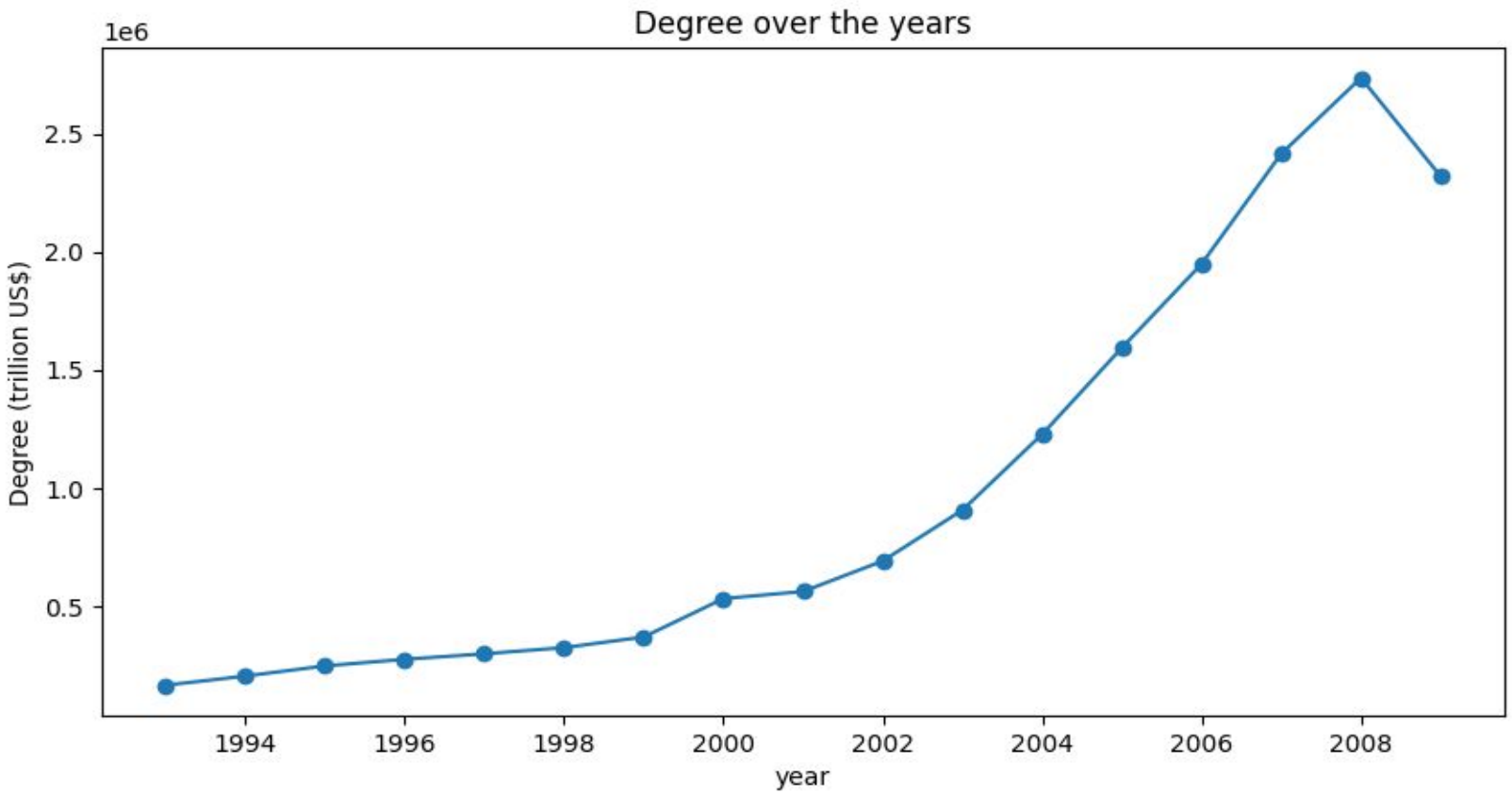
Impact of major world events

China joins WTO

Clustering Coefficient of China Over the Years



China Degree Over the Years



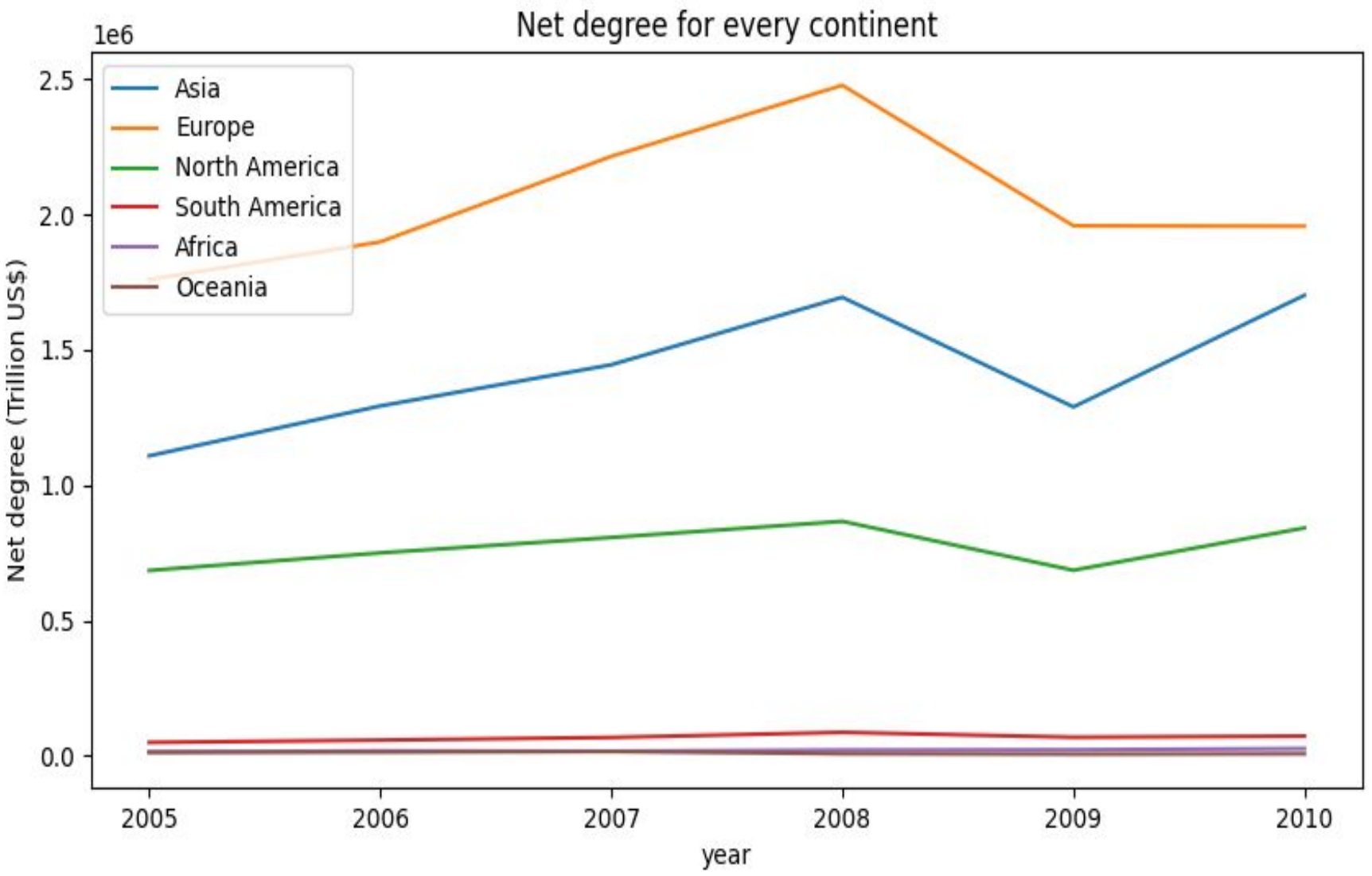
We can see a steep increase since 2001 as China entered WTO.

2008 Recession

Recession

- global economic downturn that began in the United States and quickly spread to other parts of the world.
- The crisis led to a domino effect, causing widespread bank failures, stock market crashes, and a sharp decline in consumer and investor confidence.
- We shall use the Network from our database to analyse the data

Continent-wise comparison of 2008 crisis



Impact of 2008 recession on the network