Network analysis on global trade data

Deng et al. (2021)

Social network analysis of virtual water trade among major countries in the world

In [2]:
```
import math
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

pd.set_option('display.max_columns', None)

from collections import Counter
import numpy as np
from operator import itemgetter
import powerlaw as pwl

from networkx.algorithms import community
```

Data prep

Visit the following website and download the bilateral trade data: https://www.fao.org/faostat/en/#data

- Trade > Detailed trade matrix

Find dataset for the node

- Population and Employment
- Macro Indicators
- SDG Indictors

Navigate the metadata and other resources

Load the trade data

In [7]:
```
link = pd.read_csv("D:/Trade_DetailedTradeMatrix_E_All_Data_(Normalized).csv")
```

In [214…
```
# inspect the data
link.head()
```

Out[214…

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Item	Element	Year	Unit	Value
0	2	Afghanistan	9	Argentina	Cake, oilseeds nes	Import quantity	2017	t	463.11
1	2	Afghanistan	9	Argentina	Cake, oilseeds nes	Import quantity	2019	t	1192.62
2	3	Albania	3	Albania	Cigarettes	Import quantity	2004	t	12.00
3	4	Algeria	2	Afghanistan	Almonds, shelled	Import quantity	2005	t	3.00
4	3	Albania	3	Albania	Cigarettes	Import value	2004	1000 USD	104.00

Create an edge list

The final data format: O_code | D_code | O_name | D_name | year | item | weight (traded volume)

In [217…
```
# select variables
link = link[['Reporter Country Code', 'Reporter Countries', 'Partner Country Code', 'Partner Countries', 'Item', 'Element', 'Y
```

In [218…
```
# filter down.. export quantity only.. select items... Remove 0 values
link1 = link[(link["Value"]>0) & ((link["Element"]) == ("Export quantity")) & ((link["Item"])== ("Soya beans"))]
link1
```

Out[218...

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Item	Element	Year	Unit	Value
26328	105	Israel	3	Albania	Soya beans	Export quantity	2017	t	10.91
35091	226	Uganda	4	Algeria	Soya beans	Export quantity	2015	t	74.97
63906	21	Brazil	2	Afghanistan	Soya beans	Export quantity	2020	t	4044.68
80830	230	Ukraine	2	Afghanistan	Soya beans	Export quantity	2015	t	202.55
86125	214	China, Taiwan Province of	7	Angola	Soya beans	Export quantity	2013	t	1.00
...	...	...	...	...	...	...	...	...	...
50389610	150	Netherlands (Kingdom of the)	231	United States of America	Soya beans	Export quantity	2016	t	7.94
50389611	150	Netherlands (Kingdom of the)	231	United States of America	Soya beans	Export quantity	2018	t	0.53
50389612	150	Netherlands (Kingdom of the)	231	United States of America	Soya beans	Export quantity	2020	t	0.10
50389613	150	Netherlands (Kingdom of the)	231	United States of America	Soya beans	Export quantity	2022	t	12.00
50399602	150	Netherlands (Kingdom of the)	234	Uruguay	Soya beans	Export quantity	2019	t	0.01

27219 rows × 9 columns

In [219...

```
link1["Value"].sort_values(ascending = True)
```

Out[219…

```
50399602           0.01
39065438           0.01
39065466           0.01
39069717           0.01
39250367           0.01
                  ...
17755507    57963485.74
17755697    60476103.96
17755604    60595839.60
17755414    68839892.47
17755887    74471954.17
Name: Value, Length: 27219, dtype: float64
```

In [220…

```
link2 = link[(link["Value"]>0) & ((link["Element"]) == ("Import quantity")) & ((link["Item"])== ("Soya beans"))]
link2
```

Out[220…

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Item	Element	Year	Unit	Value
2368	26	Brunei Darussalam	2	Afghanistan	Soya beans	Import quantity	2020	t	2.05
9462	16	Bangladesh	9	Argentina	Soya beans	Import quantity	2005	t	61301.00
9604	16	Bangladesh	9	Argentina	Soya beans	Import quantity	2006	t	119327.00
9739	16	Bangladesh	9	Argentina	Soya beans	Import quantity	2007	t	140340.00
9883	16	Bangladesh	9	Argentina	Soya beans	Import quantity	2015	t	160032.88
...	...	...	...	...	...	...	...	...	...
50416522	150	Netherlands (Kingdom of the)	237	Viet Nam	Soya beans	Import quantity	2016	t	0.23
50416523	150	Netherlands (Kingdom of the)	237	Viet Nam	Soya beans	Import quantity	2017	t	0.14
50416524	150	Netherlands (Kingdom of the)	237	Viet Nam	Soya beans	Import quantity	2021	t	64.77
50416525	150	Netherlands (Kingdom of the)	237	Viet Nam	Soya beans	Import quantity	2022	t	155.42
50416526	150	Netherlands (Kingdom of the)	237	Viet Nam	Soya beans	Import quantity	2023	t	0.30

29183 rows × 9 columns

In [221…

```
link2["Value"].sort_values(ascending = True)
```

Out[221…
```
3818085            0.01
24907065           0.01
33849650           0.01
47541291           0.01
36009621           0.01
                  ...
11827980    57676480.00
11828220    58146822.01
11828100    64277600.00
11827860    66084250.00
11828460    71592496.82
Name: Value, Length: 29183, dtype: float64
```

In [222…
```
link3 = link[(link["Value"]>0) & ((link["Element"]) == ("Export quantity")) & ((link["Item"])== ("beef and veal preparations n
link3
```

Out[222...

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Item	Element	Year	Unit	Value
20	9	Argentina	2	Afghanistan	beef and veal preparations nes	Export quantity	1998	t	25.00
22	9	Argentina	2	Afghanistan	beef and veal preparations nes	Export quantity	2007	t	35.00
35	9	Argentina	2	Afghanistan	beef and veal preparations nes	Export quantity	2008	t	47.00
37	9	Argentina	2	Afghanistan	beef and veal preparations nes	Export quantity	2009	t	88.00
52	9	Argentina	2	Afghanistan	beef and veal preparations nes	Export quantity	2010	t	23.00
...	...	...	...	...	...	...	...	...	...
50418329	150	Netherlands (Kingdom of the)	249	Yemen	beef and veal preparations nes	Export quantity	2021	t	18.86
50418330	150	Netherlands (Kingdom of the)	249	Yemen	beef and veal preparations nes	Export quantity	2022	t	16.94
50418331	150	Netherlands (Kingdom of the)	249	Yemen	beef and veal preparations nes	Export quantity	2023	t	34.79
50420919	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	beef and veal preparations nes	Export quantity	1991	t	34.00
50422604	150	Netherlands (Kingdom of the)	251	Zambia	beef and veal preparations nes	Export quantity	1989	t	3.00

42187 rows × 9 columns

In [223...

```
link3["Value"].sort_values(ascending = True)
```

Out[223…
```
44174853        0.01
12336535        0.01
20054670        0.01
44579978        0.01
35201174        0.01
               ...
38415652    60191.00
38821626    65221.00
38821598    66010.00
38415232    67481.00
40728195    80097.00
Name: Value, Length: 42187, dtype: float64
```

In [231…
```
link4 = link[(link["Value"]>0) & ((link["Element"]) == ("Import quantity")) & ((link["Item"])== ("beef and veal preparations n
link4
```

Out[231...

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Item	Element	Year	Unit	Value
1521	47	Cook Islands	10	Australia	beef and veal preparations nes	Import quantity	1999	t	10.00
1583	47	Cook Islands	10	Australia	beef and veal preparations nes	Import quantity	2001	t	1.00
2925	90	Guinea	2	Afghanistan	beef and veal preparations nes	Import quantity	2016	t	4.27
3406	8	Antigua and Barbuda	9	Argentina	beef and veal preparations nes	Import quantity	2005	t	18.00
3646	86	Grenada	8	Antigua and Barbuda	beef and veal preparations nes	Import quantity	2002	t	16.00
...	...	...	...	...	...	...	...	...	...
50420909	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	beef and veal preparations nes	Import quantity	1987	t	18.00
50420910	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	beef and veal preparations nes	Import quantity	1989	t	12.00
50420911	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	beef and veal preparations nes	Import quantity	1990	t	14.00
50420912	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	beef and veal preparations nes	Import quantity	1991	t	2.00
50422602	150	Netherlands (Kingdom of the)	251	Zambia	beef and veal preparations nes	Import quantity	1986	t	11.00

31974 rows × 9 columns

In [232...

```
link4["Value"].sort_values(ascending = True)
```

Out[232…
```
40116831        0.01
26903581        0.01
4451872         0.01
4447004         0.01
4404068         0.01
               ...
18122062    65876.00
20540821    66699.00
20540906    67878.00
40646756    74945.00
40646778    77288.00
Name: Value, Length: 31974, dtype: float64
```

In [233…
```
soybeans = pd.merge(
    link1,
    link2,
    on=["Reporter Country Code", "Reporter Countries", "Partner Country Code", "Partner Countries", "Year", "Item"],
    suffixes=("_x", "_y")
)

soybeans = soybeans[[
    "Reporter Country Code",
    "Reporter Countries",
    "Partner Country Code",
    "Partner Countries",
    "Year",
    "Value_x",  # Export
    "Value_y"   # Import
]]

soybeans["Item"] = "Soy beans"

soybeans["Final Value"] = soybeans[["Value_x", "Value_y"]].max(axis=1)

soybeans
```

Out[233...

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Year	Value_x	Value_y	Item	Final Value
0	18	Bhutan	100	India	2010	19.00	39.00	Soy beans	39.00
1	18	Bhutan	100	India	2012	1.00	1.00	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	2365.00	4.00	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	9298.05	80.00	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	22.17	0.96	Soy beans	22.17
...	...	...	...	...	...	...	...	...	...
8738	150	Netherlands (Kingdom of the)	231	United States of America	2016	7.94	2058590.25	Soy beans	2058590.25
8739	150	Netherlands (Kingdom of the)	231	United States of America	2018	0.53	3029938.01	Soy beans	3029938.01
8740	150	Netherlands (Kingdom of the)	231	United States of America	2020	0.10	1582196.88	Soy beans	1582196.88
8741	150	Netherlands (Kingdom of the)	231	United States of America	2022	12.00	1765343.12	Soy beans	1765343.12
8742	150	Netherlands (Kingdom of the)	234	Uruguay	2019	0.01	223165.62	Soy beans	223165.62

8743 rows × 9 columns

In [234...

```
beef = pd.merge(
    link3,
    link4,
    on=["Reporter Country Code", "Reporter Countries", "Partner Country Code", "Partner Countries", "Year", "Item"],
```

```
    suffixes=("_x", "_y")
)

beef = beef[[
    "Reporter Country Code",
    "Reporter Countries",
    "Partner Country Code",
    "Partner Countries",
    "Year",
    "Value_x",  # Export
    "Value_y",  # Import
]]

beef["Item"] = "Beef"

beef["Final Value"] = beef[["Value_x", "Value_y"]].max(axis=1)
beef
```

Out[234...

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Year	Value_x	Value_y	Item	Final Value
0	244	Samoa	10	Australia	2021	25.00	0.86	Beef	25.00
1	14	Barbados	8	Antigua and Barbuda	1992	4.00	13.00	Beef	13.00
2	168	Papua New Guinea	10	Australia	2020	0.10	3.86	Beef	3.86
3	168	Papua New Guinea	10	Australia	2021	0.01	154.71	Beef	154.71
4	154	North Macedonia	3	Albania	2017	2.33	0.03	Beef	2.33
...	...	...	...	...	...	...	...	...	...
10714	150	Netherlands (Kingdom of the)	231	United States of America	1997	1.00	11.00	Beef	11.00
10715	150	Netherlands (Kingdom of the)	231	United States of America	2021	3.43	0.19	Beef	3.43
10716	150	Netherlands (Kingdom of the)	231	United States of America	2022	2.06	3.75	Beef	3.75
10717	150	Netherlands (Kingdom of the)	234	Uruguay	2023	0.69	17.83	Beef	17.83
10718	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	1991	34.00	2.00	Beef	34.00

10719 rows × 9 columns

In [235...

```
edge = pd.concat([soybeans, beef], ignore_index = True)
edge
```

Out[235...

	Reporter Country Code	Reporter Countries	Partner Country Code	Partner Countries	Year	Value_x	Value_y	Item	Final Value
0	18	Bhutan	100	India	2010	19.00	39.00	Soy beans	39.00
1	18	Bhutan	100	India	2012	1.00	1.00	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	2365.00	4.00	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	9298.05	80.00	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	22.17	0.96	Soy beans	22.17
...	...	...	...	...	...	...	...	...	...
19457	150	Netherlands (Kingdom of the)	231	United States of America	1997	1.00	11.00	Beef	11.00
19458	150	Netherlands (Kingdom of the)	231	United States of America	2021	3.43	0.19	Beef	3.43
19459	150	Netherlands (Kingdom of the)	231	United States of America	2022	2.06	3.75	Beef	3.75
19460	150	Netherlands (Kingdom of the)	234	Uruguay	2023	0.69	17.83	Beef	17.83
19461	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	1991	34.00	2.00	Beef	34.00

19462 rows × 9 columns

In [236...

```
edge = edge.rename(columns={
    "Reporter Country Code" : "O_code",
    "Reporter Countries" : "O_name",
    "Partner Country Code": "D_code",
```

```
    "Partner Countries": "D_name",
    "Year": "year",
    "Item": "item",
    "Final Value": "weight"
})[["O_code", "O_name", "D_code", "D_name", "year", "item", "weight"]]
```

In [237…

```
edge
```

Out[237…

	O_code	O_name	D_code	D_name	year	item	weight
0	18	Bhutan	100	India	2010	Soy beans	39.00
1	18	Bhutan	100	India	2012	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	Soy beans	22.17
...	...	...	...	...	...	...	...
19457	150	Netherlands (Kingdom of the)	231	United States of America	1997	Beef	11.00
19458	150	Netherlands (Kingdom of the)	231	United States of America	2021	Beef	3.43
19459	150	Netherlands (Kingdom of the)	231	United States of America	2022	Beef	3.75
19460	150	Netherlands (Kingdom of the)	234	Uruguay	2023	Beef	17.83
19461	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	1991	Beef	34.00

19462 rows × 7 columns

In [238…

```
# save the result
edge.to_csv("edge.csv")
```

Create a node data combining multiple sources

The final data format: country | country_code | continent | population | var1 | var2 | var3...

In [240…
```
# load datasets
macro = pd.read_csv("D:/Macro-Statistics_Key_Indicators_E_All_Data_(Normalized).csv", encoding_errors = 'ignore')
pop= pd.read_csv("D:/Population_E_All_Data_(Normalized).csv", encoding_errors = 'ignore')
```

In [241…
```
macro = macro[["Area Code", "Area", "Item", "Element", "Year", "Unit", "Value"]]
```

In [242…
```
macro
```

Out[242...

	Area Code	Area	Item	Element	Year	Unit	Value
0	2	Afghanistan	Gross Domestic Product	Value Standard Local Currency	1970	million SLC	78.697146
1	2	Afghanistan	Gross Domestic Product	Value Standard Local Currency	1971	million SLC	82.397024
2	2	Afghanistan	Gross Domestic Product	Value Standard Local Currency	1972	million SLC	71.797487
3	2	Afghanistan	Gross Domestic Product	Value Standard Local Currency	1973	million SLC	77.997271
4	2	Afghanistan	Gross Domestic Product	Value Standard Local Currency	1974	million SLC	96.996607
...	...	...	...	...	...	...	...
704309	5817	Net Food Importing Developing Countries	Gross National Income	Annual growth US$ per capita	2018	%	-0.935393
704310	5817	Net Food Importing Developing Countries	Gross National Income	Annual growth US$ per capita	2019	%	-1.205496
704311	5817	Net Food Importing Developing Countries	Gross National Income	Annual growth US$ per capita	2020	%	-2.433519
704312	5817	Net Food Importing Developing Countries	Gross National Income	Annual growth US$ per capita	2021	%	9.098567
704313	5817	Net Food Importing Developing Countries	Gross National Income	Annual growth US$ per capita	2022	%	2.424958

704314 rows × 7 columns

In [243...
```
pop = pop[["Area Code", "Area", "Item", "Element", "Year", "Unit", "Value"]]
```

In [244...
```
pop
```

Out[244...

	Area Code	Area	Item	Element	Year	Unit	Value
0	2	Afghanistan	Population - Est. & Proj.	Total Population - Both sexes	1950	1000 No	7776.176
1	2	Afghanistan	Population - Est. & Proj.	Total Population - Both sexes	1951	1000 No	7879.339
2	2	Afghanistan	Population - Est. & Proj.	Total Population - Both sexes	1952	1000 No	7987.783
3	2	Afghanistan	Population - Est. & Proj.	Total Population - Both sexes	1953	1000 No	8096.698
4	2	Afghanistan	Population - Est. & Proj.	Total Population - Both sexes	1954	1000 No	8207.950
...	...	...	...	...	...	...	...
168684	5817	Net Food Importing Developing Countries	Population - Est. & Proj.	Urban population	2046	1000 No	1383146.702
168685	5817	Net Food Importing Developing Countries	Population - Est. & Proj.	Urban population	2047	1000 No	1417990.918
168686	5817	Net Food Importing Developing Countries	Population - Est. & Proj.	Urban population	2048	1000 No	1453215.998
168687	5817	Net Food Importing Developing Countries	Population - Est. & Proj.	Urban population	2049	1000 No	1488809.964
168688	5817	Net Food Importing Developing Countries	Population - Est. & Proj.	Urban population	2050	1000 No	1524760.153

168689 rows × 7 columns

In [258...
```
# merge them
```

In [260...
```
macro["Element"].unique()
```

Out[260…
```
array(['Value Standard Local Currency', 'Value US$',
       'Value US$ per capita',
       'Value Standard Local Currency, 2015 prices',
       'Value US$ per capita, 2015 prices', 'Value US$, 2015 prices',
       'Annual growth Standard Local Currency',
       'Annual growth Standard Local Currency, 2015 prices',
       'Annual growth US$', 'Annual growth US$ per capita',
       'Annual growth US$, 2015 prices',
       'Annual growth US$ per capita, 2015 prices', 'Share of GDP US$',
       'Share of GDP US$, 2015 prices',
       'Share of GDP Standard Local Currency',
       'Share of GDP Standard Local Currency, 2015 prices',
       'Ratio of Value Added (Agriculture, Forestry and Fishing) Standard Local Currency',
       'Ratio of Value Added (Agriculture, Forestry and Fishing) US$',
       'Share of Value Added (Total Manufacturing) Standard Local Currency',
       'Share of Value Added (Total Manufacturing) US$'], dtype=object)
```

In [262…
```
pop["Element"].unique()
```

Out[262…
```
array(['Total Population - Both sexes', 'Total Population - Male',
       'Total Population - Female', 'Rural population',
       'Urban population'], dtype=object)
```

In [264…
```
macro = macro[(macro["Value"]>0) & (macro["Element"] == "Share of GDP US$, 2015 prices")]
macro = macro[(macro["Area Code"] < 5000)]
```

In [266…
```
pop = pop[(pop["Value"]>0) & (pop["Element"] == "Total Population - Both sexes")]
pop = pop[(pop["Area Code"] < 5000) & (pop["Year"] <= 2023)]
```

In [268…
```
macro = macro.rename(columns={"Area": "country", "Area Code": "country_code", "Year": "year", "Value": "GDP"})
macro = macro[["country", "country_code", "year", "GDP"]]

pop = pop.rename(columns={"Area": "country", "Area Code": "country_code", "Year": "year", "Value": "Population"})
pop = pop[["country", "country_code", "year", "Population"]]
```

In [270…
```
node = pd.merge(macro, pop, on=["country", "country_code", "year"], how= "inner")

print(node.head())
print(node.tail())
```

```
       country  country_code  year        GDP  Population
0  Afghanistan             2  1970  16.015601   11290.128
1  Afghanistan             2  1971  17.130539   11567.667
2  Afghanistan             2  1972  17.929952   11853.696
3  Afghanistan             2  1973  17.425341   12157.999
4  Afghanistan             2  1974  21.245487   12469.127
        country  country_code  year       GDP  Population
31797  Zimbabwe           181  2009  3.024533   13142.790
31798  Zimbabwe           181  2010  3.519815   13356.548
31799  Zimbabwe           181  2011  3.606485   13595.424
31800  Zimbabwe           181  2012  3.603453   13817.887
31801  Zimbabwe           181  2013  3.874855   14013.808
```

In [272…
```
# save the result
node.to_csv("node.csv")
```

In [274…
```
edge.head()
```

Out[274…

	O_code	O_name	D_code	D_name	year	item	weight
0	18	Bhutan	100	India	2010	Soy beans	39.00
1	18	Bhutan	100	India	2012	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	Soy beans	22.17

In [276…
```
# check if the countries match well with the edge list above 링크에 있는 컨트리가 노드에있는지랑 노드에 있는 컨트리가 링크에 있
node_countries = set(node["country"].unique())
origin_countries = set(edge["O_name"].unique())
dest_countries = set(edge["D_name"].unique())

missing_in_node_O = origin_countries - node_countries
print("Reporter Countries' name that are not in node:", missing_in_node_O)

missing_in_node_D = dest_countries - node_countries
print("Partner Countries' name that are not in node:", missing_in_node_D)
```

```
Reporter Countries' name that are not in node: {'Réunion', "Côte d'Ivoire", 'Türkiye', 'Serbia and Montenegro', 'Belgium-Luxemb
ourg', 'China, Taiwan Province of', 'French Guiana'}
Partner Countries' name that are not in node: {'Réunion', "Côte d'Ivoire", 'Türkiye', 'Serbia and Montenegro', 'Belgium-Luxembo
urg', 'China, Taiwan Province of', 'Guadeloupe', 'French Guiana'}
```

Analysis

- density, asymmetry
- out-degree, in-degree (fitting power-law distribution)
- network visualization
- temporal change
- alternative visualizations: chordal graph

In [279…

```
# directional edge/ value, column for the weight
# density for each year, save the information, create a plot
# use a for loop
```

In [281…

```
edge.head()
```

Out[281…

	O_code	O_name	D_code	D_name	year	item	weight
0	18	Bhutan	100	India	2010	Soy beans	39.00
1	18	Bhutan	100	India	2012	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	Soy beans	22.17

In [283…

```
node.head()
```

Out[283…

	country	country_code	year	GDP	Population
0	Afghanistan	2	1970	16.015601	11290.128
1	Afghanistan	2	1971	17.130539	11567.667
2	Afghanistan	2	1972	17.929952	11853.696
3	Afghanistan	2	1973	17.425341	12157.999
4	Afghanistan	2	1974	21.245487	12469.127

In [285…

```
G = nx.from_pandas_edgelist(edge, source = 'O_name', target = 'D_name',
                            edge_attr = 'weight', create_using = nx.DiGraph()
                           )
```

In [287…

```
node_s = node.drop_duplicates(subset='country')
node_attr = node_s.set_index('country').to_dict('index')
```

In [289…

```
edge_s = edge[edge["year"] == 2022]
edge_s
```

Out[289...

	O_code	O_name	D_code	D_name	year	item	weight
24	149	Nepal	33	Canada	2022	Soy beans	20089.58
27	33	Canada	4	Algeria	2022	Soy beans	278261.30
32	256	Luxembourg	11	Austria	2022	Soy beans	0.38
36	144	Mozambique	41	China, mainland	2022	Soy beans	615.66
58	19	Bolivia (Plurinational State of)	9	Argentina	2022	Soy beans	450057.11
...	...	...	...	...	...	...	...
19379	150	Netherlands (Kingdom of the)	211	Switzerland	2022	Beef	9.59
19419	68	France	229	United Kingdom of Great Britain and Northern I...	2022	Beef	2433.22
19421	68	France	231	United States of America	2022	Beef	0.02
19455	150	Netherlands (Kingdom of the)	229	United Kingdom of Great Britain and Northern I...	2022	Beef	971.27
19459	150	Netherlands (Kingdom of the)	231	United States of America	2022	Beef	3.75

1041 rows × 7 columns

In [291...

```
density = nx.density(G)
print("Network density:", density)
```

```
Network density: 0.06232153941651148
```

In [293...

```
year = sorted(edge["year"].drop_duplicates())
year
```

Out[293…
```
[1986,
 1987,
 1988,
 1989,
 1990,
 1991,
 1992,
 1993,
 1994,
 1995,
 1996,
 1997,
 1998,
 1999,
 2000,
 2001,
 2002,
 2003,
 2004,
 2005,
 2006,
 2007,
 2008,
 2009,
 2010,
 2011,
 2012,
 2013,
 2014,
 2015,
 2016,
 2017,
 2018,
 2019,
 2020,
 2021,
 2022,
 2023]
```

In [295…
```
min(year)
```

Out[295… 1986

Density

Global Index (density, 1986-2023)

In [718…
```
y1 = []
densities_s =[]
for i in range(len(year)):
    y = i + min(year)
    soybeans_s = soybeans[soybeans["Year"] == y]

    node_s = node_s.drop_duplicates(subset='country')
    node_s = node_s[node_s["year"] == y]
    node_attr_soy = node_s.set_index('country').to_dict('index')

    G_s = nx.from_pandas_edgelist(soybeans_s, source = 'Reporter Countries', target = 'Partner Countries',
                            edge_attr = 'Final Value', create_using = nx.DiGraph()
                           )

    in_degree_dict_s = dict(G_s.in_degree(weight='weight'))
    out_degree_dict_s = dict(G_s.out_degree(weight='weight'))


    nx.set_node_attributes(G_s, node_attr)

    density_s = nx.density(G_s)

    y1.append(y)
    densities_s.append(density_s)
```

In [720…
```
y1 = []
densities_b =[]
for i in range(len(year)):
    y = i + min(year)
    beef_s = beef[beef["Year"] == y]

    node_s = node_s.drop_duplicates(subset='country')
```

```
    node_s = node_s[node_s["year"] == y]
    node_attr_beef = node_s.set_index('country').to_dict('index')

    G_b = nx.from_pandas_edgelist(beef_s, source = 'Reporter Countries', target = 'Partner Countries',
                            edge_attr = 'Final Value', create_using = nx.DiGraph()
                           )

    in_degree_dict_b = dict(G_b.in_degree(weight='weight'))
    out_degree_dict_b = dict(G_b.out_degree(weight='weight'))


    nx.set_node_attributes(G_b, node_attr)

    density_b = nx.density(G_b)

    y1.append(y)
    densities_b.append(density_b)
```

In [574…

```
y1
```

Out[574…
```
[1986,
 1987,
 1988,
 1989,
 1990,
 1991,
 1992,
 1993,
 1994,
 1995,
 1996,
 1997,
 1998,
 1999,
 2000,
 2001,
 2002,
 2003,
 2004,
 2005,
 2006,
 2007,
 2008,
 2009,
 2010,
 2011,
 2012,
 2013,
 2014,
 2015,
 2016,
 2017,
 2018,
 2019,
 2020,
 2021,
 2022,
 2023]
```

In [576…
```
print(densities_s)
```

```
[0.05291005291005291, 0.0582010582010582, 0.06417112299465241, 0.058823529411764705, 0.05930930930930931, 0.0761904761904762,
0.05708245243128964, 0.055697278911564625, 0.05061224489795919, 0.03565166569257744, 0.04188311688311688, 0.04204476709013914,
0.04350282485875706, 0.03763440860215054, 0.04415653093601269, 0.04689265536723164, 0.036871270247229325, 0.03797190517998244,
0.038576300085251494, 0.040144596651445964, 0.03843226788432268, 0.04161490683229813, 0.04033485540334855, 0.04916593503072871,
0.0494824016563147, 0.04192982456140351, 0.04208860759493671, 0.05496870109546166, 0.04979570990806946, 0.046153846153846156,
0.0506155950752394, 0.04669652855543113, 0.04524840239125953, 0.046072974644403214, 0.043689320388349516, 0.042452043369474564,
0.047979797979797977, 0.05036512027491409]
```

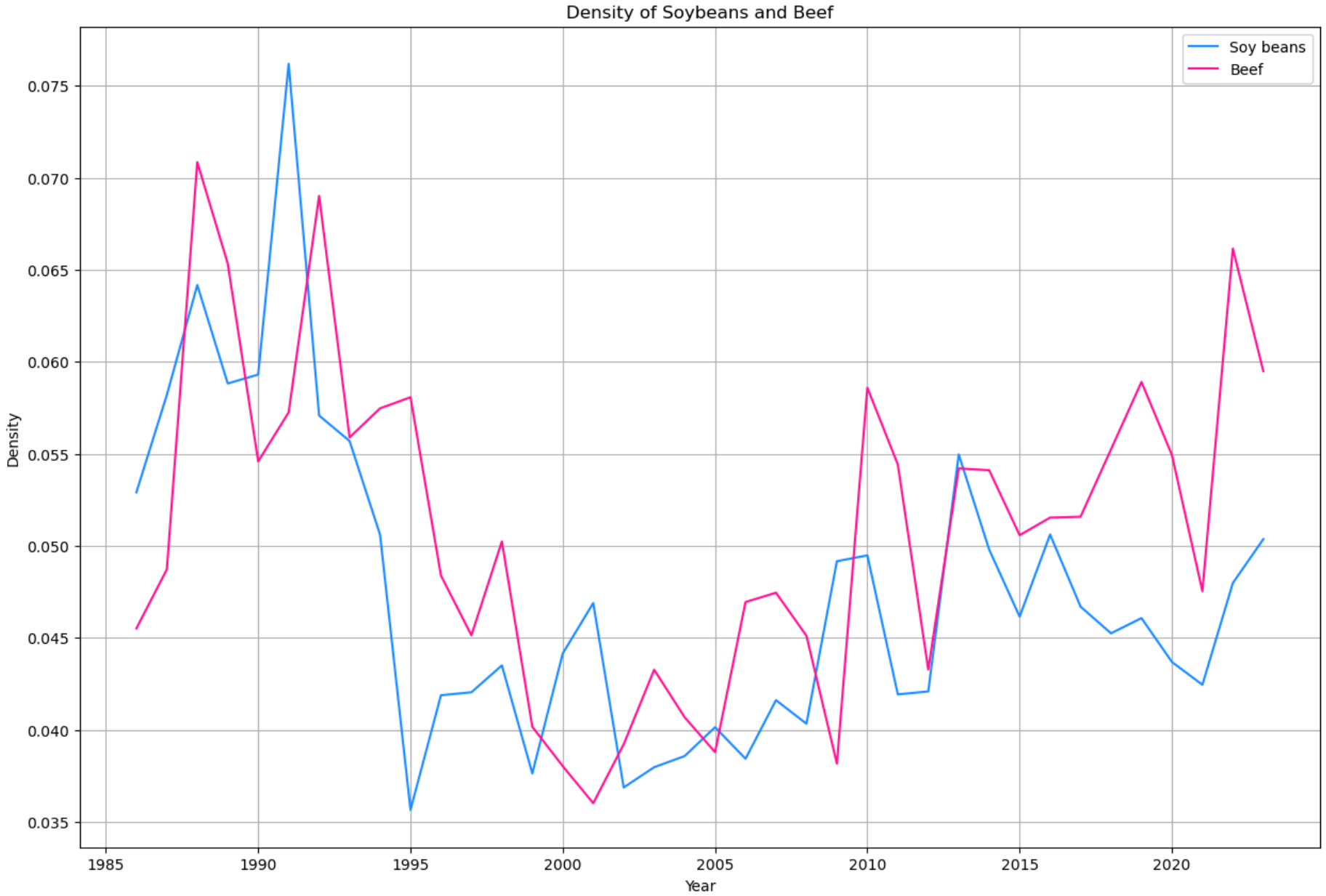
In [577…

```
print(densities_b)
```

```
[0.04551282051282051, 0.04871794871794872, 0.0708502024291498, 0.06533776301218161, 0.05458937198067633, 0.05725490196078432,
0.06901960784313725, 0.05589225589225589, 0.05747126436781609, 0.05807622504537205, 0.04838709677419355, 0.04513888888888889,
0.05023796932839767, 0.04016681299385426, 0.038028169014084505, 0.03601609657947686, 0.0392156862745098, 0.04326923076923077,
0.0406885758998435, 0.03879015721120984, 0.046948356807511735, 0.047452547452547456, 0.0451152223304122, 0.03816689466484268,
0.05859213250517598, 0.05442428730099963, 0.04327731092436975, 0.05421052631578947, 0.05410893454371715, 0.050579896907216496,
0.051535087719298246, 0.05158415841584158, 0.05524634096062667, 0.05890603085553997, 0.054937126365697794, 0.0475396163469558,
0.06615240766713418, 0.05949312714776632]
```

In [724…

```
plt.figure(figsize=(15,10))
plt.plot(y1, densities_s, label = "Soy beans", color='dodgerblue')
plt.plot(y1, densities_b, label = "Beef", color='deeppink')
plt.title("Density of Soybeans and Beef")
plt.xlabel("Year")
plt.ylabel("Density")
plt.legend()
plt.grid(True)
plt.savefig('density.png')
```



In Degree, Out Degree

In [582...]
```
edge
```

Out[582...]

	O_code	O_name	D_code	D_name	year	item	weight
0	18	Bhutan	100	India	2010	Soy beans	39.00
1	18	Bhutan	100	India	2012	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	Soy beans	22.17
...	...	...	...	...	...	...	...
19457	150	Netherlands (Kingdom of the)	231	United States of America	1997	Beef	11.00
19458	150	Netherlands (Kingdom of the)	231	United States of America	2021	Beef	3.43
19459	150	Netherlands (Kingdom of the)	231	United States of America	2022	Beef	3.75
19460	150	Netherlands (Kingdom of the)	234	Uruguay	2023	Beef	17.83
19461	150	Netherlands (Kingdom of the)	248	Yugoslav SFR	1991	Beef	34.00

19462 rows × 7 columns

In [584...]
```
y1 = []
for i in range(len(year)):

    y = i + min(year)
    edge_s = edge[edge["year"] == y]

    node_s = node_s.drop_duplicates(subset='country')
    node_s = node_s[node_s["year"] == y]
    node_attr = node_s.set_index('country').to_dict('index')

    G = nx.from_pandas_edgelist(edge_s, source = 'O_name', target = 'D_name',
                                edge_attr = 'weight', create_using = nx.DiGraph()
                               )
```

```
    nx.set_node_attributes(G, node_attr)

    out_degree_dict = dict(G.out_degree(weight='Value'))


    in_degree_dict = dict(G.in_degree(weight='Value'))
    out_degree_dict = dict(G.out_degree(weight='Value'))


    nx.set_node_attributes(G, out_degree_dict, 'in_degree')
    nx.set_node_attributes(G, in_degree_dict, 'out_degree')

    sorted(out_degree_dict.items(), key=itemgetter(1), reverse=True)
    sorted(in_degree_dict.items(), key=itemgetter(1), reverse=True)

print(out_degree_dict)
```

```
{'Togo': 4, 'Belgium': 24, 'Canada': 23, 'Algeria': 0, 'Luxembourg': 9, 'Austria': 25, 'Bolivia (Plurinational State of)': 3,
'Argentina': 6, 'Paraguay': 5, 'Serbia': 18, 'China, mainland': 4, 'Burkina Faso': 1, "Côte d'Ivoire": 2, 'Uruguay': 4, 'Ukrain
e': 14, 'Ethiopia': 1, 'China, Taiwan Province of': 6, 'Australia': 7, 'Slovakia': 12, 'United Republic of Tanzania': 2, 'Roman
ia': 22, 'Mozambique': 3, 'Malawi': 3, 'Croatia': 16, 'Kazakhstan': 5, 'Denmark': 20, 'Brazil': 4, 'China, Macao SAR': 1, 'Chin
a, Hong Kong SAR': 6, 'Ghana': 5, 'Slovenia': 11, 'Poland': 20, 'Bosnia and Herzegovina': 5, 'Zambia': 2, 'India': 4, 'Czechi
a': 20, 'United States of America': 22, 'Eswatini': 1, 'South Africa': 7, 'New Zealand': 6, 'Hungary': 15, 'Chile': 0, 'Sri Lan
ka': 2, 'Türkiye': 12, 'Sweden': 21, 'Switzerland': 8, 'United Arab Emirates': 13, 'Botswana': 3, 'Spain': 20, 'France': 36, 'B
ulgaria': 13, 'Zimbabwe': 1, 'Estonia': 8, 'Finland': 5, 'Latvia': 8, 'Guyana': 2, 'Armenia': 1, 'Russian Federation': 0, 'Neth
erlands (Kingdom of the)': 35, 'Italy': 26, 'Germany': 32, 'Thailand': 7, 'Cambodia': 0, 'Peru': 3, 'Ecuador': 0, 'Lithuania':
10, 'Greece': 19, 'Guatemala': 3, 'Honduras': 0, 'United Kingdom of Great Britain and Northern Ireland': 22, 'Japan': 5, 'Costa
Rica': 3, 'Cameroon': 0, 'Malaysia': 8, 'Republic of Moldova': 2, 'Singapore': 9, 'Tajikistan': 0, 'Norway': 7, 'Portugal': 9,
'Indonesia': 4, 'Ireland': 16, 'Cyprus': 2, 'Saudi Arabia': 0, 'Iran (Islamic Republic of)': 0, 'Yemen': 0, 'Republic of Kore
a': 0, 'Mexico': 1, 'Nepal': 0, 'Turkmenistan': 0, 'Suriname': 0, 'Viet Nam': 0, 'Mauritius': 0, 'Morocco': 3, 'Tunisia': 0, 'M
alta': 0, 'Senegal': 0, 'Uganda': 0, 'Fiji': 2, 'Philippines': 7, 'Barbados': 0, 'Djibouti': 1, 'Georgia': 2, 'Azerbaijan': 3,
'Namibia': 3, 'Uzbekistan': 2, 'North Macedonia': 2, 'Brunei Darussalam': 1, 'Cabo Verde': 1, 'Colombia': 0, 'El Salvador': 1,
'Jamaica': 2, 'Kyrgyzstan': 1, 'Montenegro': 1, 'Kuwait': 2, 'Jordan': 4, 'Lebanon': 2, 'Bahrain': 3, 'Egypt': 0, 'Oman': 2, 'Q
atar': 0, 'Maldives': 1, 'Iceland': 0}
```

In [587…

```
# Out-degree (edges going *out* of the node)
out_degree_dict = dict(G.out_degree(weight='weight'))
```

```
# In-degree (edges coming *into* the node)
in_degree_dict = dict(G.in_degree(weight='Value'))
```

In [589…

```
nx.set_node_attributes(G, out_degree_dict, 'i_degree')
nx.set_node_attributes(G, in_degree_dict, 'o_degree')
```

In [590…

```
sorted(out_degree_dict.items(), key=itemgetter(1), reverse=True)
```

Out[590…

```
[('United States of America', 31053585.990000002),
 ('China, mainland', 29335077.16),
 ('Argentina', 6388092.75),
 ('Paraguay', 6222362.25),
 ('Brazil', 4607914.899999999),
 ('Canada', 3747086.29),
 ('Spain', 1544461.8299999998),
 ('Ukraine', 1041603.01),
 ('Thailand', 443567.58999999997),
 ('Bolivia (Plurinational State of)', 371615.97000000003),
 ('Malaysia', 299019.57),
 ('Portugal', 265725.48),
 ('Costa Rica', 250611.88),
 ('United Kingdom of Great Britain and Northern Ireland', 240095.73),
 ('Germany', 219744.72),
 ('Italy', 191826.16),
 ('Uruguay', 190138.08),
 ('Türkiye', 181073.76),
 ('Poland', 178270.09999999998),
 ('Romania', 162246.08000000002),
 ('Belgium', 152898.16999999998),
 ('France', 138353.48),
 ('Netherlands (Kingdom of the)', 137906.26),
 ('Peru', 93060.90000000001),
 ('United Republic of Tanzania', 76937.48000000001),
 ('China, Taiwan Province of', 64985.19),
 ('Togo', 42152.17),
 ('Austria', 38123.58),
 ('Ireland', 37699.89),
 ('Zambia', 36865.92),
 ('Ghana', 26114.14),
 ('Japan', 22369.57),
 ('Denmark', 20249.88),
 ('Zimbabwe', 17068.2),
 ('Sweden', 15199.01),
 ('Bosnia and Herzegovina', 14638.53),
 ('Czechia', 14395.369999999999),
 ('Australia', 13724.33),
 ('Republic of Moldova', 12957.55),
 ('Hungary', 12649.75),
```

```
('Kazakhstan', 10108.919999999998),
('Switzerland', 9175.52),
('Serbia', 8270.55),
('Burkina Faso', 7877.0),
('New Zealand', 7608.05),
('South Africa', 7407.32),
('Ethiopia', 7062.0),
("Côte d'Ivoire", 6706.120000000001),
('Slovakia', 6671.4800000000005),
('Slovenia', 6226.29),
('China, Hong Kong SAR', 5706.410000000001),
('Jordan', 4222.32),
('Malawi', 3707.68),
('Philippines', 3236.4300000000003),
('United Arab Emirates', 3207.77),
('Singapore', 3126.4700000000003),
('Indonesia', 2869.1499999999996),
('Croatia', 2814.28),
('Estonia', 2251.3),
('Finland', 2003.78),
('Lithuania', 1205.1200000000001),
('Namibia', 811.73),
('Guatemala', 732.97),
('Luxembourg', 725.29),
('Bulgaria', 674.82),
('Greece', 655.83),
('Kuwait', 563.56),
('Azerbaijan', 522.38),
('Lebanon', 490.76),
('Mexico', 476.54),
('Oman', 465.2),
('Eswatini', 456.44),
('India', 408.53),
('Bahrain', 402.09000000000003),
('Latvia', 354.26),
('Mozambique', 310.84),
('Guyana', 252.31),
('Botswana', 233.20999999999998),
('Cyprus', 220.1),
('Fiji', 217.81),
('Norway', 160.45),
```

```
('Georgia', 132.09),
('Armenia', 126.3),
('China, Macao SAR', 119.83),
('North Macedonia', 110.74),
('Montenegro', 102.2),
('Jamaica', 75.52),
('Cabo Verde', 51.11),
('Djibouti', 26.59),
('Brunei Darussalam', 23.41),
('Uzbekistan', 23.19),
('Sri Lanka', 11.5),
('Morocco', 10.17),
('Maldives', 9.1),
('El Salvador', 6.92),
('Kyrgyzstan', 1.5),
('Algeria', 0),
('Chile', 0),
('Russian Federation', 0),
('Cambodia', 0),
('Ecuador', 0),
('Honduras', 0),
('Cameroon', 0),
('Tajikistan', 0),
('Saudi Arabia', 0),
('Iran (Islamic Republic of)', 0),
('Yemen', 0),
('Republic of Korea', 0),
('Nepal', 0),
('Turkmenistan', 0),
('Suriname', 0),
('Viet Nam', 0),
('Mauritius', 0),
('Tunisia', 0),
('Malta', 0),
('Senegal', 0),
('Uganda', 0),
('Barbados', 0),
('Colombia', 0),
('Egypt', 0),
('Qatar', 0),
('Iceland', 0)]
```

In [593…

```
sorted(in_degree_dict.items(), key=itemgetter(1), reverse=True)
```

Out[593…

```
[('United States of America', 35),
 ('France', 32),
 ('Germany', 31),
 ('Italy', 28),
 ('Poland', 24),
 ('Netherlands (Kingdom of the)', 24),
 ('Belgium', 23),
 ('Austria', 23),
 ('Czechia', 23),
 ('Spain', 23),
 ('Sweden', 22),
 ('United Kingdom of Great Britain and Northern Ireland', 22),
 ('Romania', 20),
 ('Denmark', 19),
 ('Ireland', 17),
 ('Serbia', 16),
 ('China, mainland', 16),
 ('Hungary', 15),
 ('Croatia', 14),
 ('Ukraine', 13),
 ('Slovakia', 13),
 ('Lithuania', 13),
 ('Canada', 12),
 ('Slovenia', 12),
 ('Greece', 12),
 ('China, Taiwan Province of', 11),
 ('Bulgaria', 11),
 ('Japan', 11),
 ('Estonia', 10),
 ('Latvia', 10),
 ('Republic of Korea', 10),
 ('Portugal', 9),
 ('Luxembourg', 8),
 ('Australia', 8),
 ('Chile', 8),
 ('Türkiye', 8),
 ('New Zealand', 7),
 ('Switzerland', 7),
 ('United Arab Emirates', 7),
 ('Philippines', 7),
```

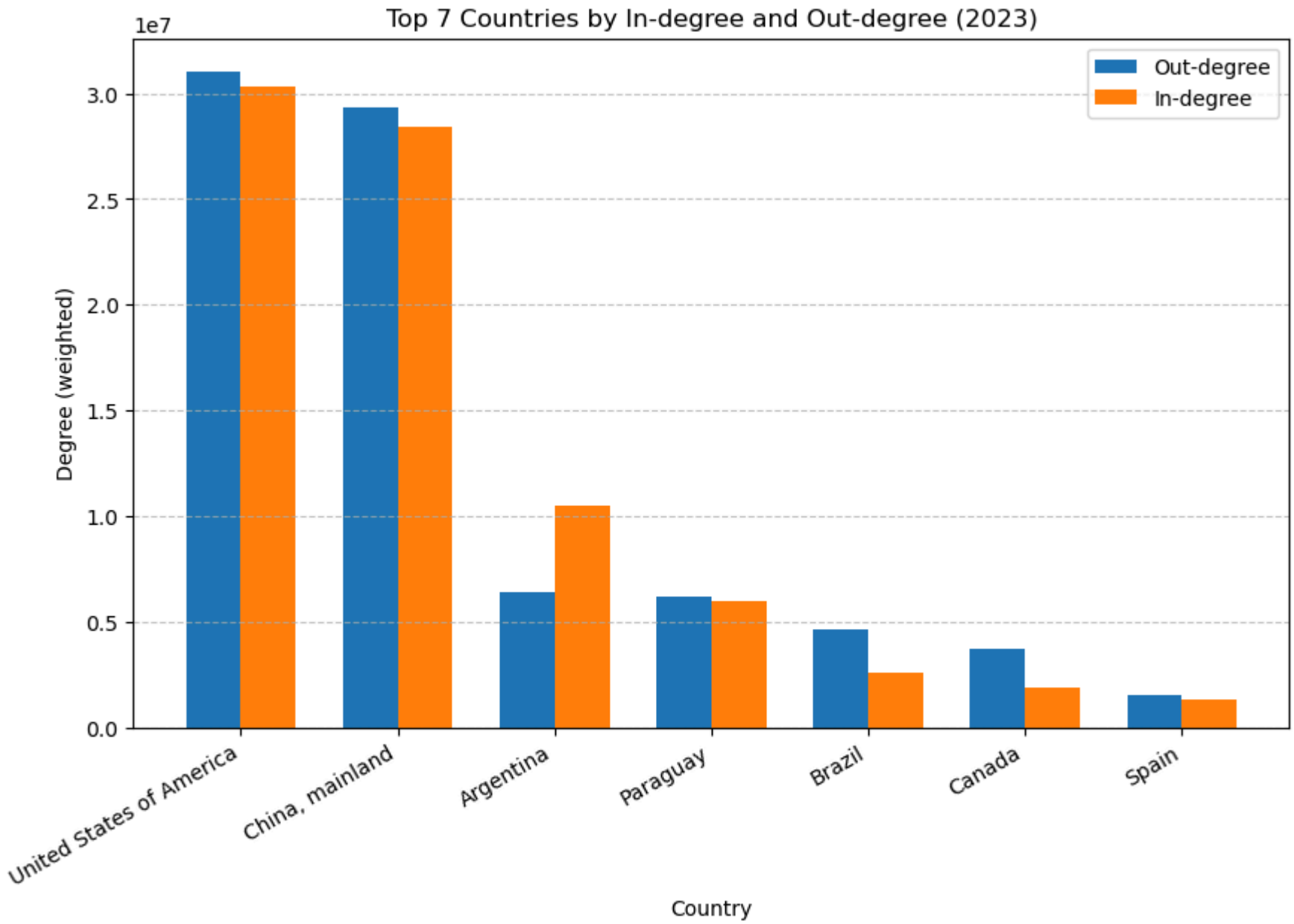
```
('Argentina', 6),
('China, Hong Kong SAR', 6),
('South Africa', 6),
('Russian Federation', 6),
('Brazil', 5),
('India', 5),
('Malaysia', 5),
('Norway', 5),
('Uruguay', 4),
('Bosnia and Herzegovina', 4),
('Zimbabwe', 4),
('Thailand', 4),
('Singapore', 4),
('Indonesia', 4),
('Cyprus', 4),
('Viet Nam', 4),
('Jordan', 4),
('Paraguay', 3),
('Ghana', 3),
('Zambia', 3),
('Finland', 3),
('Saudi Arabia', 3),
('Mexico', 3),
('Malta', 3),
('Bolivia (Plurinational State of)', 2),
("Côte d'Ivoire", 2),
('Botswana', 2),
('Guatemala', 2),
('Honduras', 2),
('Costa Rica', 2),
('Uzbekistan', 2),
('North Macedonia', 2),
('Lebanon', 2),
('Iceland', 2),
('Algeria', 1),
('Burkina Faso', 1),
('United Republic of Tanzania', 1),
('Malawi', 1),
('Kazakhstan', 1),
('China, Macao SAR', 1),
('Eswatini', 1),
```

```
('Sri Lanka', 1),
('Cambodia', 1),
('Ecuador', 1),
('Cameroon', 1),
('Republic of Moldova', 1),
('Tajikistan', 1),
('Iran (Islamic Republic of)', 1),
('Yemen', 1),
('Nepal', 1),
('Turkmenistan', 1),
('Suriname', 1),
('Mauritius', 1),
('Morocco', 1),
('Tunisia', 1),
('Senegal', 1),
('Uganda', 1),
('Fiji', 1),
('Barbados', 1),
('Georgia', 1),
('Azerbaijan', 1),
('Namibia', 1),
('Brunei Darussalam', 1),
('Colombia', 1),
('Kyrgyzstan', 1),
('Montenegro', 1),
('Kuwait', 1),
('Egypt', 1),
('Qatar', 1),
('Togo', 0),
('Ethiopia', 0),
('Mozambique', 0),
('Guyana', 0),
('Armenia', 0),
('Peru', 0),
('Djibouti', 0),
('Cabo Verde', 0),
('El Salvador', 0),
('Jamaica', 0),
('Bahrain', 0),
('Oman', 0),
('Maldives', 0)]
```

In [595…
```
edge_2023 = edge[edge["year"] == 2023]

node_y = node_s[node_s["year"] == y].drop_duplicates(subset='country')
node_attr = node_y.set_index('country').to_dict('index')

G_2023 = nx.from_pandas_edgelist(edge_2023, source='O_name', target='D_name',
                                 edge_attr='weight', create_using=nx.DiGraph())

nx.set_node_attributes(G_2023, node_attr)

out_degree_dict = dict(G_2023.out_degree(weight='weight'))
in_degree_dict = dict(G_2023.in_degree(weight='weight'))

top7_out_2023 = sorted(out_degree_dict.items(), key=itemgetter(1), reverse=True)[:7]
top7_in_2023 = sorted(in_degree_dict.items(), key=itemgetter(1), reverse=True)[:7]

print(top7_out_2023)
```

```
[('United States of America', 31053585.990000002), ('China, mainland', 29335077.16), ('Argentina', 6388092.75), ('Paraguay', 62
22362.25), ('Brazil', 4607914.899999999), ('Canada', 3747086.29), ('Spain', 1544461.8299999998)]
```

In [726…
```
out_labels, out_values = zip(*top7_out_2023)
in_labels, in_values = zip(*top7_in_2023)
# Plot
plt.figure(figsize=(10, 6))
bar_width = 0.35
x = range(len(out_labels))
plt.bar(x, out_values, width=bar_width, label='Out-degree', align='center')
plt.bar([p + bar_width for p in x], in_values, width=bar_width, label='In-degree', align='center')
plt.xlabel('Country')
plt.ylabel('Degree (weighted)')
plt.title('Top 7 Countries by In-degree and Out-degree (2023)')
plt.xticks([p + bar_width / 2 for p in x], out_labels, rotation=30, ha='right')
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
plt.savefig('bar_in_out_degree.png')
```

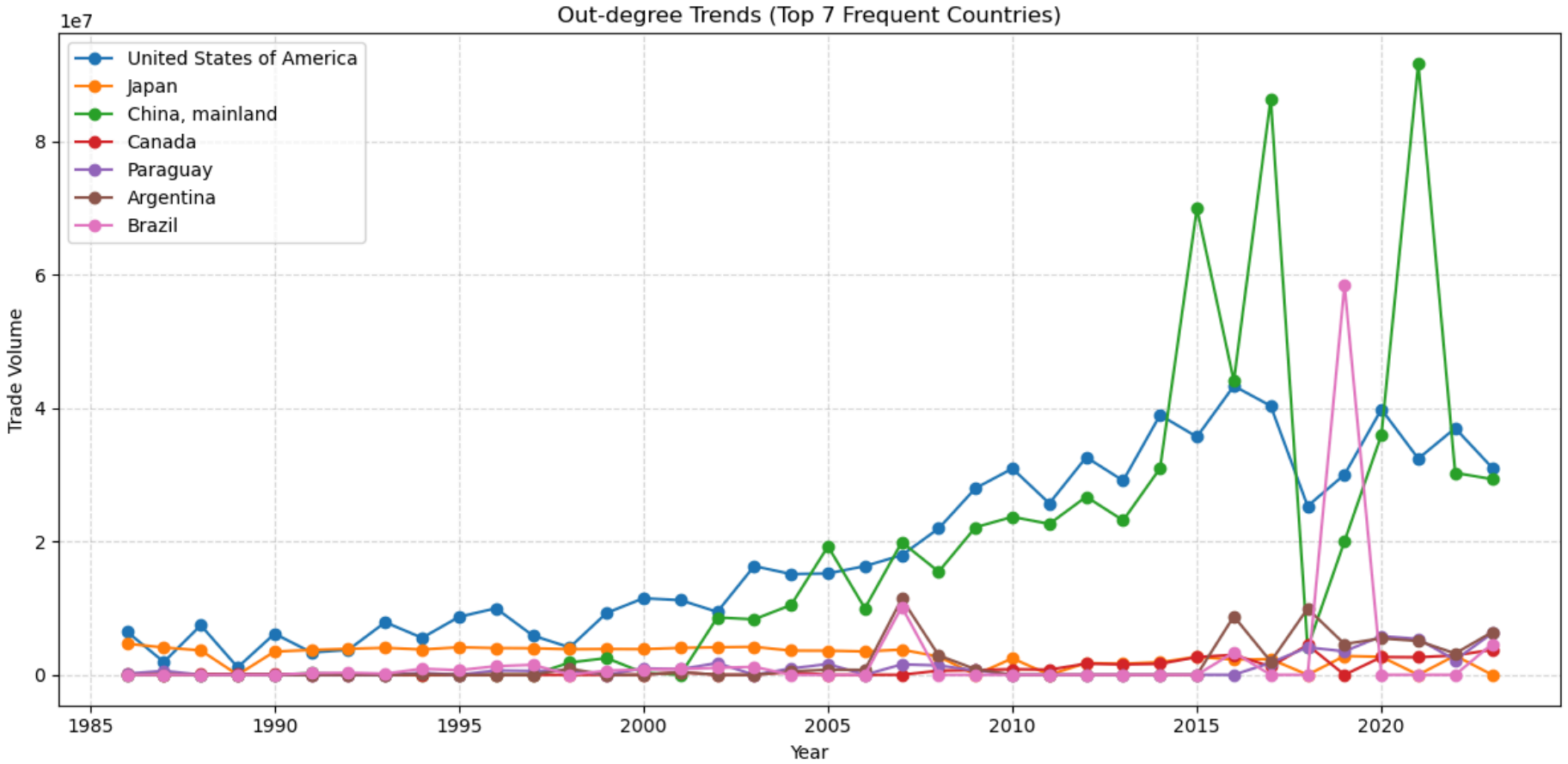



```
<Figure size 640x480 with 0 Axes>
```

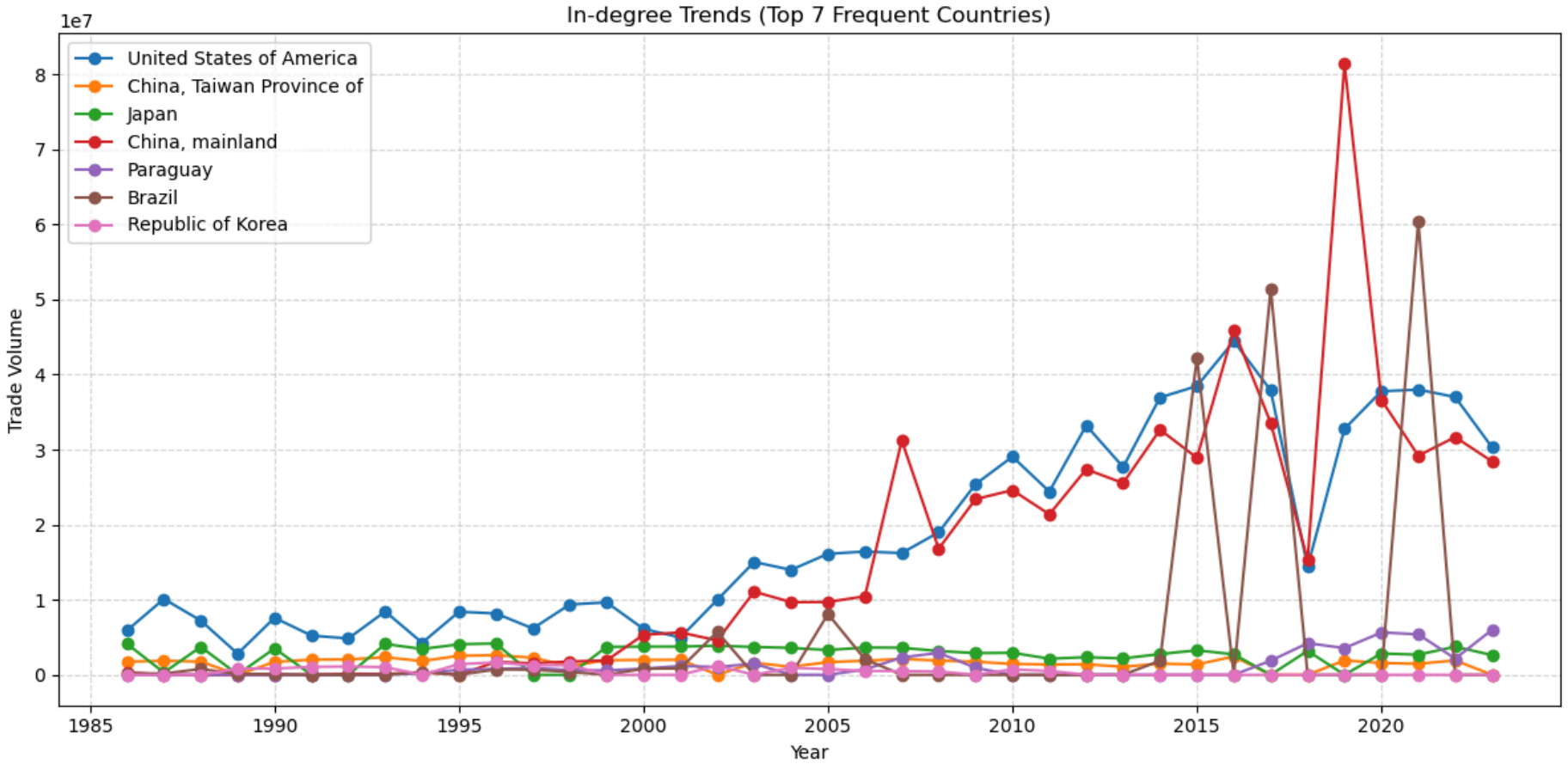
In [599…
```
top_outdegree_per_year = {}
top_indegree_per_year = {}

for i in range(len(year)):
    y = i + min(year)
    edge_s = edge[edge["year"] == y]

    node_y = node_s[node_s["year"] == y].drop_duplicates(subset='country')
    node_attr = node_y.set_index('country').to_dict('index')

    G = nx.from_pandas_edgelist(edge_s, source='O_name', target='D_name',
                                edge_attr='weight', create_using=nx.DiGraph())

    nx.set_node_attributes(G, node_attr)

    out_degree_dict = dict(G.out_degree(weight='weight'))
    in_degree_dict = dict(G.in_degree(weight='weight'))

    top7_out = sorted(out_degree_dict.items(), key=itemgetter(1), reverse=True)[:7]
    top7_in = sorted(in_degree_dict.items(), key=itemgetter(1), reverse=True)[:7]

    top_outdegree_per_year[y] = top7_out
    top_indegree_per_year[y] = top7_in
```

In [728…
```
from collections import defaultdict
# The most frequently been on top 7
def get_top_n_countries(top_dict, n=7):
    counter = defaultdict(int)
    for year_data in top_dict.values():
        for country, _ in year_data:
            counter[country] += 1
    sorted_countries = sorted(counter.items(), key=lambda x: -x[1])
    return [country for country, _ in sorted_countries[:n]]

def timeseries_dict(top_dict, top_countries):
    result = {country: [] for country in top_countries}
    years = sorted(top_dict.keys())
    for year in years:
        year_data = dict(top_dict[year])
        for country in top_countries:
```

```
            result[country].append(year_data.get(country, 0))
    return result, years

def plot_trend_line(country_dict, years, title):
    plt.figure(figsize=(12, 6))
    for country, values in country_dict.items():
        plt.plot(years, values, marker='o', label=country)
    plt.xlabel("Year")
    plt.ylabel("Trade Volume")
    plt.title(title)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.tight_layout()
    plt.show()
    plt.savefig('top7_in_out_deg.png')

top_out_countries = get_top_n_countries(top_outdegree_per_year)
top_in_countries = get_top_n_countries(top_indegree_per_year)

out_series, years = timeseries_dict(top_outdegree_per_year, top_out_countries)
in_series, _ = timeseries_dict(top_indegree_per_year, top_in_countries)

plot_trend_line(out_series, years, "Out-degree Trends (Top 7 Frequent Countries)")
plot_trend_line(in_series, years, "In-degree Trends (Top 7 Frequent Countries)")
```




```
<Figure size 640x480 with 0 Axes>
```

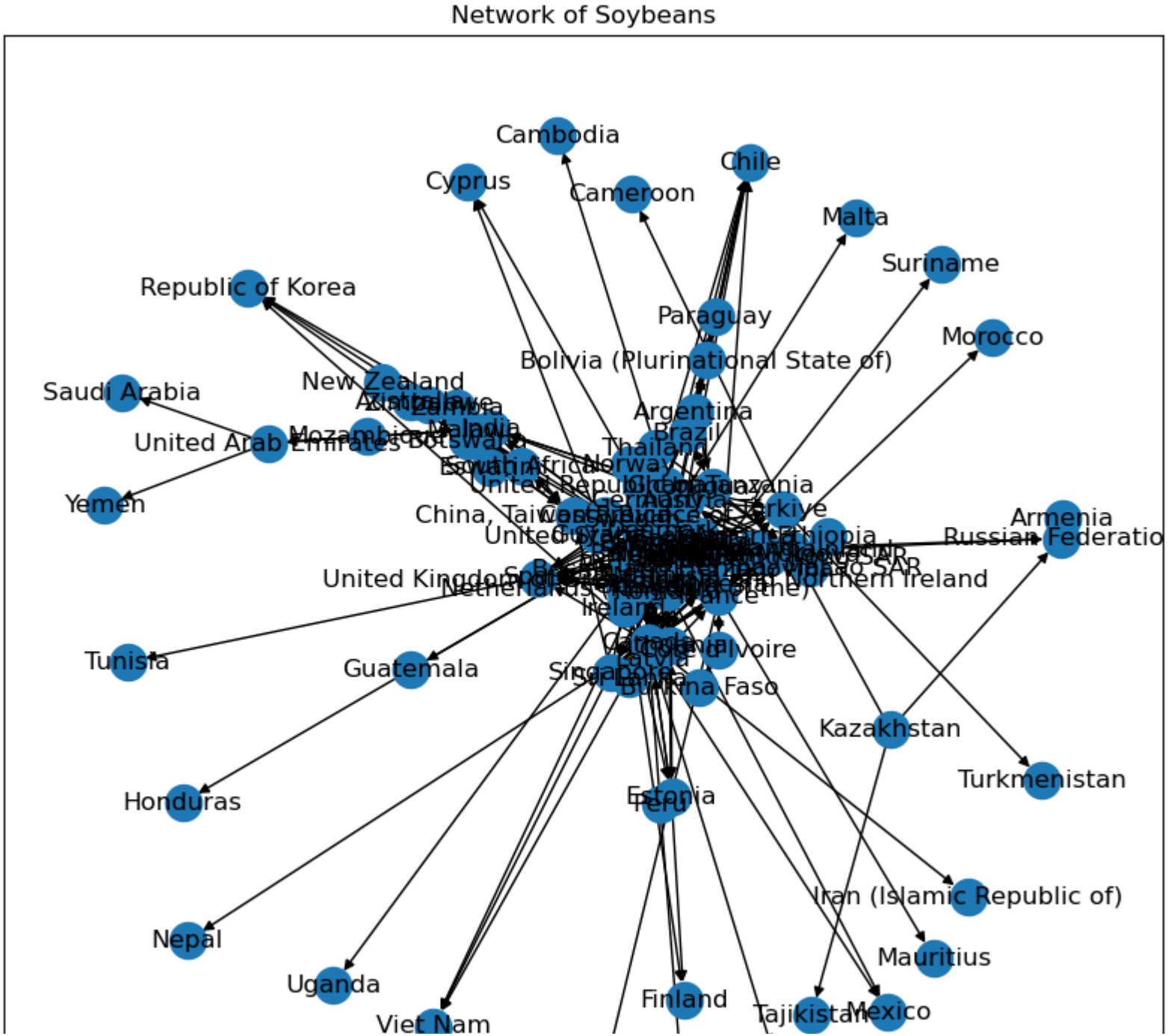


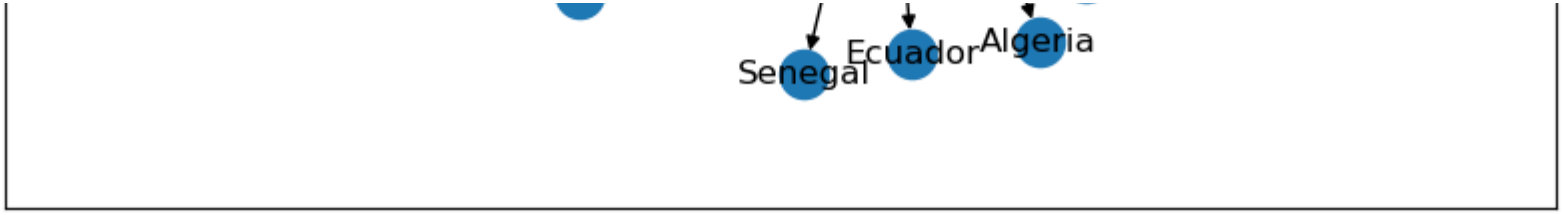

```
<Figure size 640x480 with 0 Axes>
```

Visualization

In [730…

```
plt.figure(figsize = (10,10))
plt.title("Network of Soybeans")
nx.draw_networkx(G_s, with_labels=True)
plt.show()
plt.savefig('Network_of_soybean.png')
```


Network of Soybeans
Cambodia
Chile
Cyprus
Cameroon
Malta
Suriname
Republic of Korea
Paraguay
Morocco
Bolivia (Plurinational State of)
Saudi Arabia
New Zealand
Argentina
Brazil
United Arab Emirates
Yemen
Armenia
Russian Federatio
Tunisia
Guatemala
Kazakhstan
Turkmenistan
Honduras
Estonia
Peru
Iran (Islamic Republic of)
Nepal
Mauritius
Uganda
Finland
Tajikistan
Mexico
Viet Nam



<Figure size 640x480 with 0 Axes>

In [734…
```
betweenness_dict_s = nx.betweenness_centrality(G_s) # Run betweenness centrality
eigenvector_dict_s = nx.eigenvector_centrality(G_s) # Run eigenvector centrality: How influential a node is in a network.

# Assign each to an attribute in your network
nx.set_node_attributes(G_s, betweenness_dict_s, 'betweenness_soybean')
nx.set_node_attributes(G_s, eigenvector_dict_s, 'eigenvector_soybean')
```

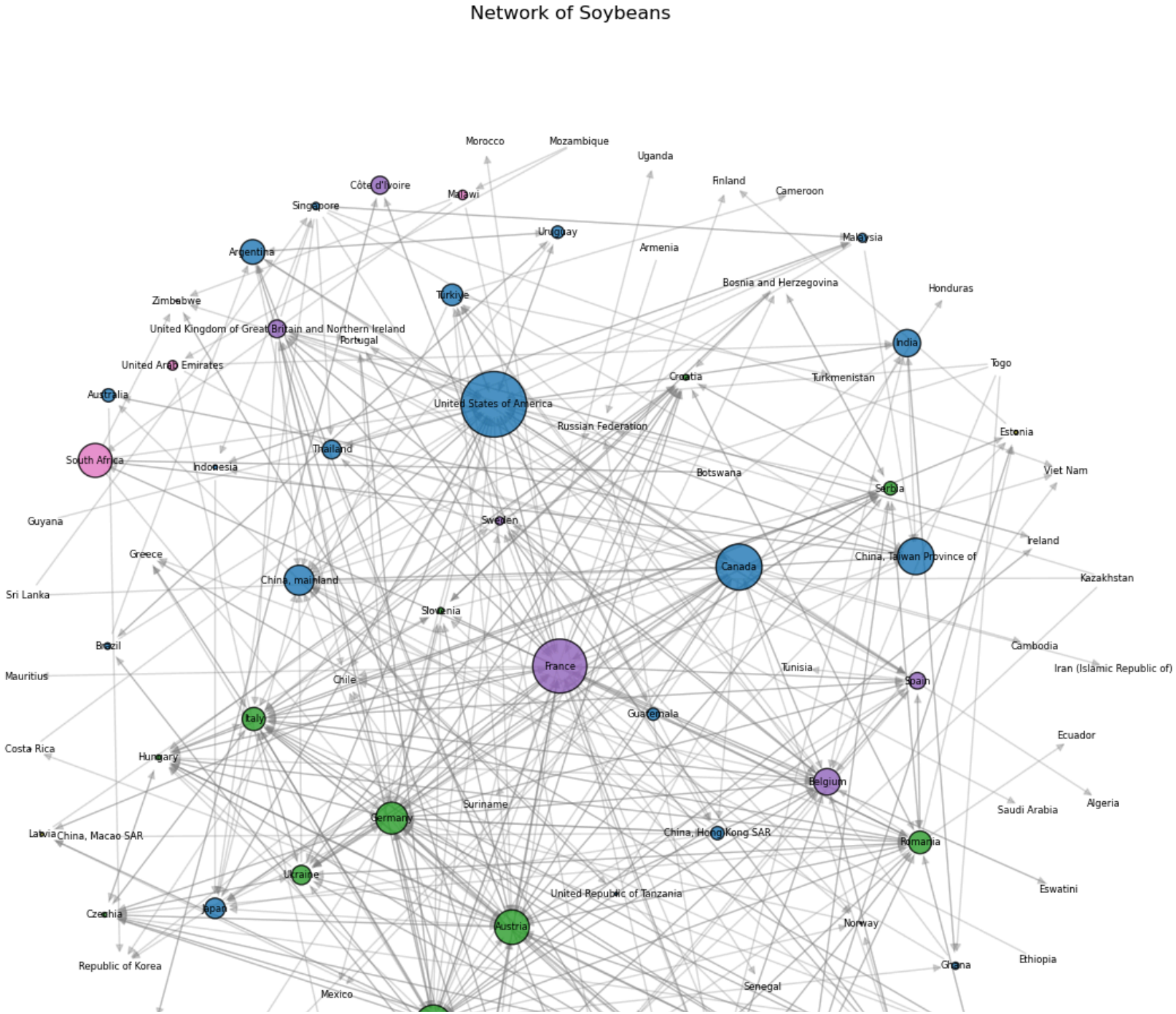
In [736…
```
communities_s = community.greedy_modularity_communities(G_s)

modularity_dict_s = {} # Create a blank dictionary
for i,c in enumerate(communities_s): # Loop through the list of communities
    for name in c: # Loop through each person in a community
        modularity_dict_s[name] = i # Create an entry in the dictionary for the person

print(modularity_dict_s)
```

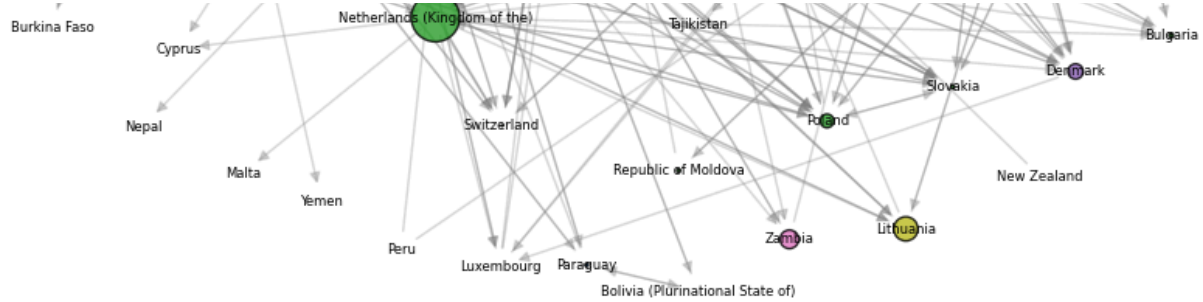
```
{'India': 0, 'United States of America': 0, 'Costa Rica': 0, 'Viet Nam': 0, 'Türkiye': 0, 'Ethiopia': 0, 'Thailand': 0, 'Canad
a': 0, 'Singapore': 0, 'China, mainland': 0, 'Ghana': 0, 'Brazil': 0, 'Ecuador': 0, 'Indonesia': 0, 'Guatemala': 0, 'Sri Lank
a': 0, 'New Zealand': 0, 'Algeria': 0, 'Nepal': 0, 'Bolivia (Plurinational State of)': 0, 'Uruguay': 0, 'Suriname': 0, 'United
Republic of Tanzania': 0, 'Australia': 0, 'Togo': 0, 'Cambodia': 0, 'Paraguay': 0, 'Turkmenistan': 0, 'Republic of Korea': 0,
'Iran (Islamic Republic of)': 0, 'Malaysia': 0, 'Guyana': 0, 'Cameroon': 0, 'Honduras': 0, 'China, Hong Kong SAR': 0, 'Japan':
0, 'China, Taiwan Province of': 0, 'China, Macao SAR': 0, 'Mexico': 0, 'Chile': 0, 'Argentina': 0, 'Peru': 0, 'Bulgaria': 1, 'P
oland': 1, 'Serbia': 1, 'Netherlands (Kingdom of the)': 1, 'Greece': 1, 'Croatia': 1, 'Italy': 1, 'Romania': 1, 'Uganda': 1, 'S
witzerland': 1, 'Ukraine': 1, 'Austria': 1, 'Germany': 1, 'Cyprus': 1, 'Slovakia': 1, 'Malta': 1, 'Republic of Moldova': 1, 'Cz
echia': 1, 'Hungary': 1, 'Slovenia': 1, 'Bosnia and Herzegovina': 1, 'Burkina Faso': 2, 'France': 2, 'Mauritius': 2, 'Senegal':
2, 'Luxembourg': 2, 'Sweden': 2, 'Denmark': 2, 'United Kingdom of Great Britain and Northern Ireland': 2, 'Spain': 2, "Côte d'I
voire": 2, 'Belgium': 2, 'Norway': 2, 'Morocco': 2, 'Tunisia': 2, 'Portugal': 2, 'Ireland': 2, 'Botswana': 3, 'Mozambique': 3,
'Yemen': 3, 'Malawi': 3, 'Eswatini': 3, 'United Arab Emirates': 3, 'Saudi Arabia': 3, 'South Africa': 3, 'Zambia': 3, 'Zimbabw
e': 3, 'Estonia': 4, 'Lithuania': 4, 'Finland': 4, 'Latvia': 4, 'Kazakhstan': 5, 'Tajikistan': 5, 'Armenia': 5, 'Russian Federa
tion': 5}
```

In [738…
```
nx.set_node_attributes(G_s, modularity_dict_s, 'modularity_soybean')
```

In [740…
```
# Extract node attributes #pop macro 활용해서 그룹화해서 색깔 지정
node_sizes = [8000 * G_s.nodes[node]['betweenness_soybean'] for node in G_s.nodes()]  # Scale betweenness
node_colors = [G_s.nodes[node]['modularity_soybean'] for node in G_s.nodes()]  # Modularity-based color

# Define colormap for modularity
cmap = plt.cm.get_cmap('tab10', max(node_colors) + 1)  # Adjust based on modularity classes

# Draw the network
plt.figure(figsize=(15, 15))
pos = nx.arf_layout(G_s, pos=None, scaling=1, a=1.1, etol=1e-06, dt=0.001, max_iter=1000)  # Use a force-directed layout for b

# Draw nodes with size and color
nx.draw_networkx_nodes(G_s, pos,
                       node_size=node_sizes,
                       node_color=node_colors,
                       cmap=cmap,
                       alpha=0.8,
                       edgecolors='black')

# Draw edges
nx.draw_networkx_edges(G_s, pos, alpha=0.3, edge_color='gray')

# Draw labels
nx.draw_networkx_labels(G_s, pos, font_size=6)


# Show the plot
plt.axis("off")
plt.title("Network of Soybeans")
plt.show()
plt.savefig('arf_of_soybeans.png')
```

```
C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\2504928339.py:6: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``mat
plotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```


Network of Soybeans
Morocco
Mozambique
Uganda
Côte d'Ivoire
Malawi
Finland
Cameroon
Singapore
Uruguay
Malaysia
Argentina
Armenia
Bosnia and Herzegovina
Honduras
Zimbabwe
Türkiye
United Kingdom of Great Britain and Northern Ireland
Portugal
India
United Arab Emirates
Togo
Croatia
Turkmenistan
Australia
United States of America
Russian Federation
Estonia
Thailand
South Africa
Indonesia
Botswana
Viet Nam
Serbia
Guyana
Sweden
Ireland
Greece
China, Taiwan Province of
Canada
China, mainland
Kazakhstan
Sri Lanka
Slovenia
Cambodia
Brazil
Iran (Islamic Republic of)
Mauritius
France
Tunisia
Chile
Spain
Guatemala
Italy
Ecuador
Costa Rica
Hungary
Belgium
Saudi Arabia
Algeria
Suriname
Germany
Latvia
China, Macao SAR
China, Hong Kong SAR
Romania
Ukraine
Eswatini
United Republic of Tanzania
Czechia
Japan
Austria
Norway
Republic of Korea
Ghana
Ethiopia
Senegal
Mexico




```
<Figure size 640x480 with 0 Axes>
```

In [342…
```
edge.head()
```

Out[342…

	O_code	O_name	D_code	D_name	year	item	weight
0	18	Bhutan	100	India	2010	Soy beans	39.00
1	18	Bhutan	100	India	2012	Soy beans	1.00
2	130	Malawi	20	Botswana	2014	Soy beans	2365.00
3	217	Togo	255	Belgium	2023	Soy beans	9298.05
4	120	Lao People's Democratic Republic	41	China, mainland	2016	Soy beans	22.17

In [344…
```
top30_out = sorted(out_degree_dict.items(), key=itemgetter(1), reverse=True)[:30]
top30_in = sorted(in_degree_dict.items(), key=itemgetter(1), reverse=True)[:30]
top30_out
```

Out[344…
```
[('United States of America', 31053585.990000002),
 ('China, mainland', 29335077.16),
 ('Argentina', 6388092.75),
 ('Paraguay', 6222362.25),
 ('Brazil', 4607914.899999999),
 ('Canada', 3747086.29),
 ('Spain', 1544461.8299999998),
 ('Ukraine', 1041603.01),
 ('Thailand', 443567.58999999997),
 ('Bolivia (Plurinational State of)', 371615.97000000003),
 ('Malaysia', 299019.57),
 ('Portugal', 265725.48),
 ('Costa Rica', 250611.88),
 ('United Kingdom of Great Britain and Northern Ireland', 240095.73),
 ('Germany', 219744.72),
 ('Italy', 191826.16),
 ('Uruguay', 190138.08),
 ('Türkiye', 181073.76),
 ('Poland', 178270.09999999998),
 ('Romania', 162246.08000000002),
 ('Belgium', 152898.16999999998),
 ('France', 138353.48),
 ('Netherlands (Kingdom of the)', 137906.26),
 ('Peru', 93060.90000000001),
 ('United Republic of Tanzania', 76937.48000000001),
 ('China, Taiwan Province of', 64985.19),
 ('Togo', 42152.17),
 ('Austria', 38123.58),
 ('Ireland', 37699.89),
 ('Zambia', 36865.92)]
```

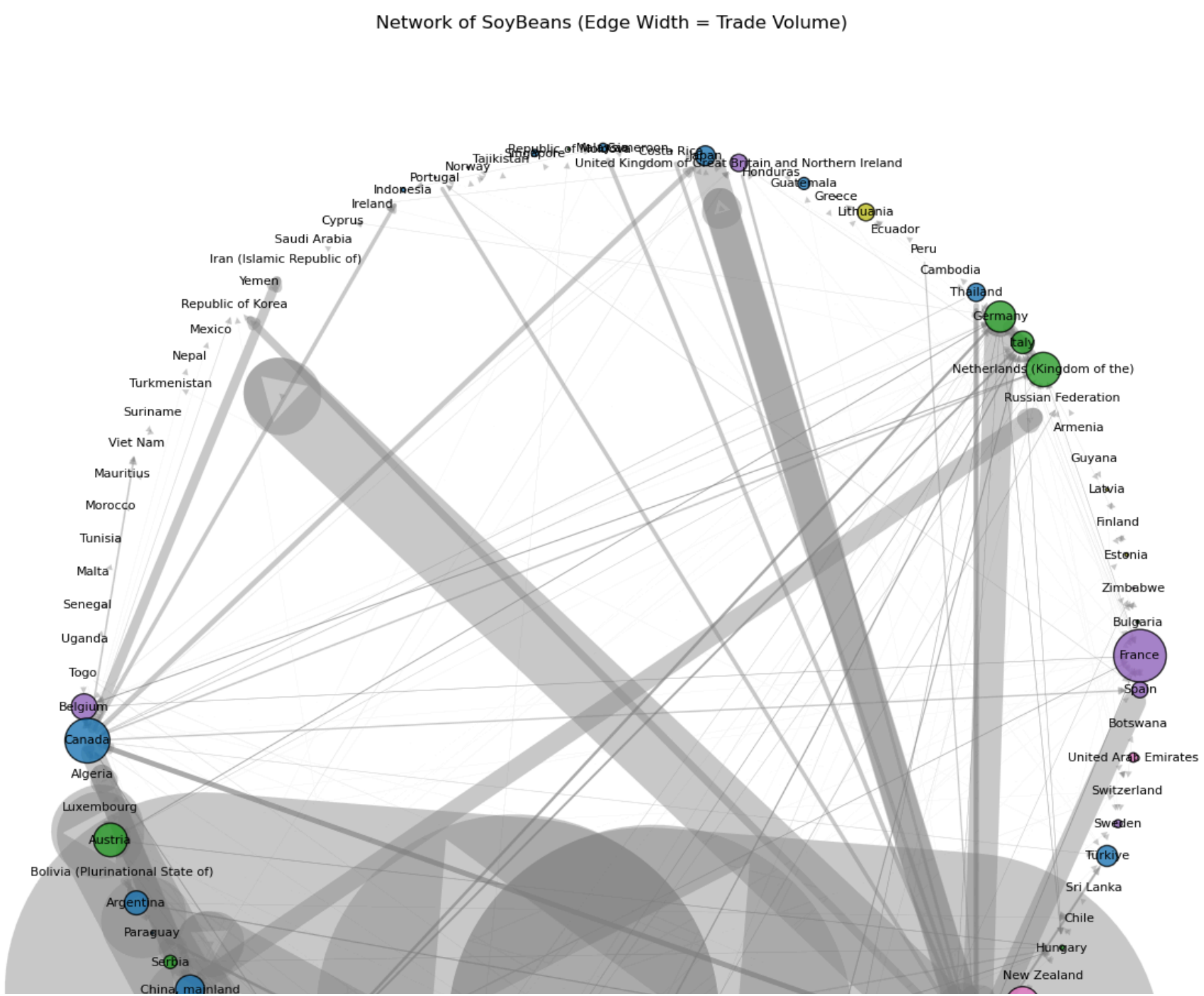
In [346…
```
soybean_30country = []

for i in range(len(top30_out)):
    country = top30_out[i][0]
    soybean_30country.append(country)

soybean_30country
```

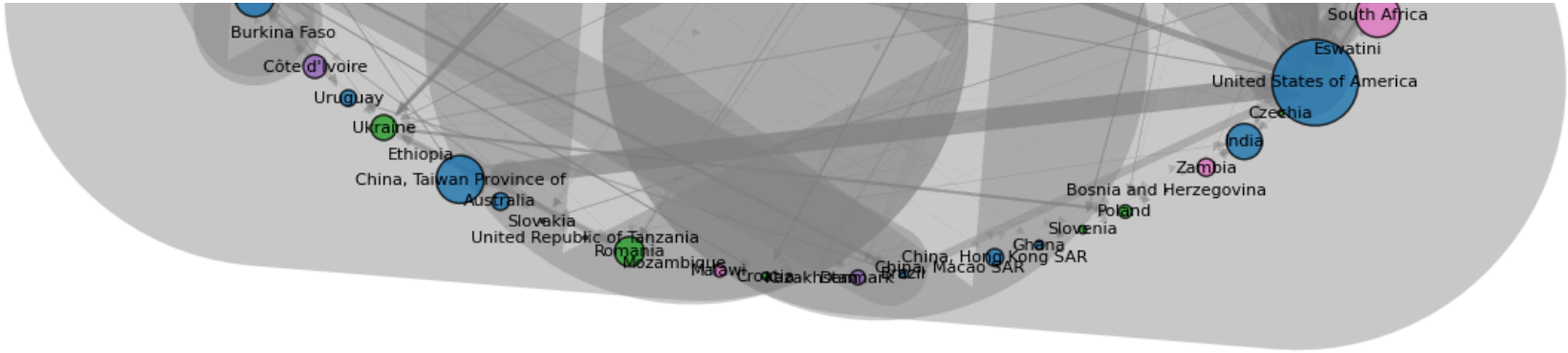
Out[346…
```
['United States of America',
 'China, mainland',
 'Argentina',
 'Paraguay',
 'Brazil',
 'Canada',
 'Spain',
 'Ukraine',
 'Thailand',
 'Bolivia (Plurinational State of)',
 'Malaysia',
 'Portugal',
 'Costa Rica',
 'United Kingdom of Great Britain and Northern Ireland',
 'Germany',
 'Italy',
 'Uruguay',
 'Türkiye',
 'Poland',
 'Romania',
 'Belgium',
 'France',
 'Netherlands (Kingdom of the)',
 'Peru',
 'United Republic of Tanzania',
 'China, Taiwan Province of',
 'Togo',
 'Austria',
 'Ireland',
 'Zambia']
```

In [744…
```python
# Extract node attributes
node_sizes = [8000 * G_s.nodes[node]['betweenness_soybean'] for node in G_s.nodes()]  # Scale betweenness
node_colors = [G_s.nodes[node]['modularity_soybean'] for node in G_s.nodes()]  # Modularity-based color

# Define colormap for modularity
cmap = plt.cm.get_cmap('tab10', max(node_colors) + 1)  # Adjust based on modularity classes

# Edge weight
edge_weights = [G_s[u][v]['Final Value'] * 0.0000098 for u, v in G_s.edges()]
```

```
# Draw the network
plt.figure(figsize=(15, 15))
pos = nx.shell_layout(G_s, nlist=None, rotate=None, scale=1, center=None, dim=2)

# Draw nodes with size and color
nx.draw_networkx_nodes(G_s, pos,
                       node_size=node_sizes,
                       node_color=node_colors,
                       cmap=cmap,
                       alpha=0.8,
                       edgecolors='black')

# Draw edges with custom width
nx.draw_networkx_edges(G_s, pos,
                       width=edge_weights,
                       alpha=0.4,
                       edge_color='gray')

# Draw labels
nx.draw_networkx_labels(G_s, pos, font_size=8)

# Show the plot
plt.axis("off")
plt.title("Network of SoyBeans (Edge Width = Trade Volume)")
plt.show()
plt.savefig('Trade_volume_of_soybeans.png')
```

```
C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\787828063.py:6: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``mat
plotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```


Network of SoyBeans (Edge Width = Trade Volume)
Tajikistan
Norway
Portugal
Indonesia
Ireland
Cyprus
Saudi Arabia
Iran (Islamic Republic of)
Yemen
Republic of Korea
Mexico
Nepal
Turkmenistan
Suriname
Viet Nam
Mauritius
Morocco
Tunisia
Malta
Senegal
Uganda
Togo
Belgium
Canada
Algeria
Luxembourg
Austria
Bolivia (Plurinational State of)
Argentina
Paraguay
Serbia
China, mainland
Costa Rica
Japan
United Kingdom of Great Britain and Northern Ireland
Honduras
Guatemala
Greece
Lithuania
Ecuador
Peru
Cambodia
Thailand
Germany
Italy
Netherlands (Kingdom of the)
Russian Federation
Armenia
Guyana
Latvia
Finland
Estonia
Zimbabwe
Bulgaria
France
Spain
Botswana
United Arab Emirates
Switzerland
Sweden
Türkiye
Sri Lanka
Chile
Hungary
New Zealand



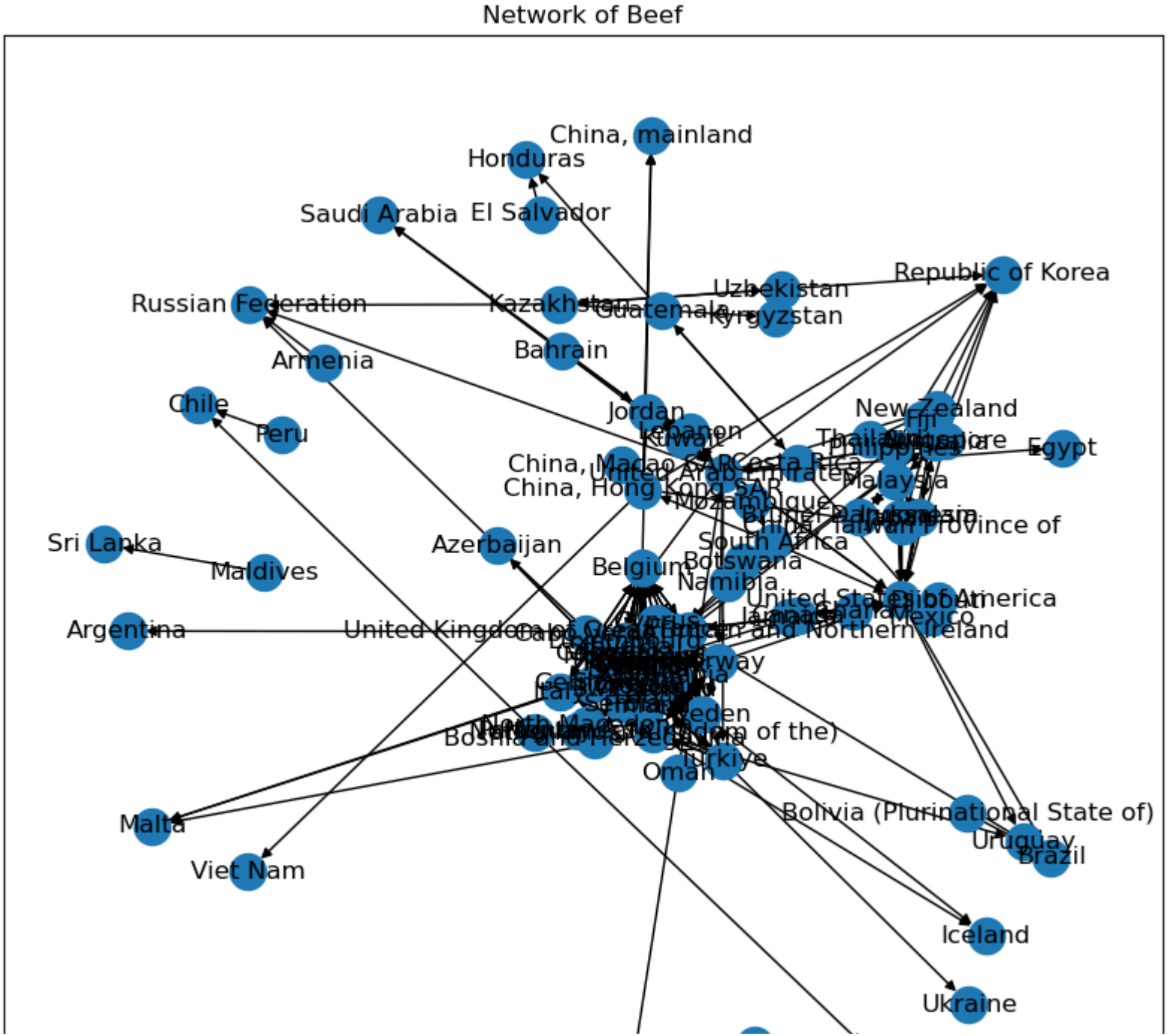

```
<Figure size 640x480 with 0 Axes>
```

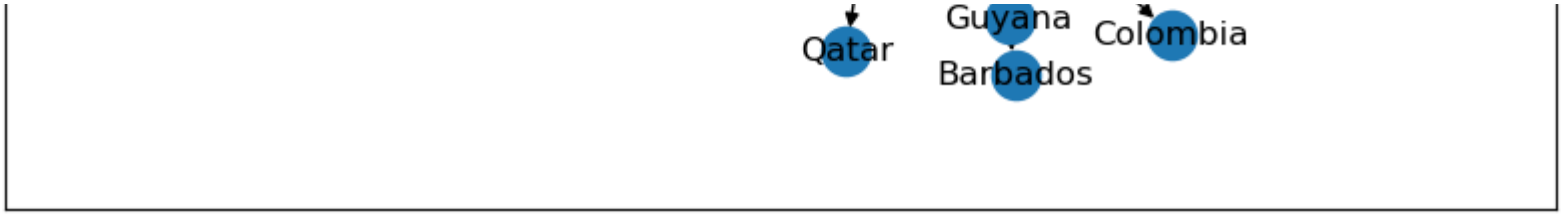
In [350…
```
node.head()
```

Out[350…

	country	country_code	year	GDP	Population
0	Afghanistan	2	1970	16.015601	11290.128
1	Afghanistan	2	1971	17.130539	11567.667
2	Afghanistan	2	1972	17.929952	11853.696
3	Afghanistan	2	1973	17.425341	12157.999
4	Afghanistan	2	1974	21.245487	12469.127

In [746…
```
plt.figure(figsize = (10,10))
plt.title("Network of Beef")
nx.draw_networkx(G_b, with_labels=True)
plt.show()
plt.savefig('Network_of_beef.png')
```


Network of Beef
China, mainland
Honduras
Saudi Arabia
El Salvador
Republic of Korea
Uzbekistan
Russian Federation
Kyrgyzstan
Bahrain
Armenia
Chile
Peru
Jordan
New Zealand
Egypt
Malaysia
Sri Lanka
Azerbaijan
South Africa
Botswana
Maldives
Belgium
Namibia
Argentina
Oman
Malta
Viet Nam
Bolivia (Plurinational State of)
Uruguay
Brazil
Iceland
Ukraine



<Figure size 640x480 with 0 Axes>

In [748…
```
betweenness_dict = nx.betweenness_centrality(G_b) # Run betweenness centrality
eigenvector_dict = nx.eigenvector_centrality(G_b) # Run eigenvector centrality: How influential a node is in a network.

# Assign each to an attribute in your network
nx.set_node_attributes(G_b, betweenness_dict, 'betweenness')
nx.set_node_attributes(G_b, eigenvector_dict, 'eigenvector')
```

In [750…
```
communities = community.greedy_modularity_communities(G_b)

modularity_dict = {} # Create a blank dictionary
for i,c in enumerate(communities): # Loop through the list of communities
    for name in c: # Loop through each person in a community
        modularity_dict[name] = i # Create an entry in the dictionary for the person

print(modularity_dict)
```

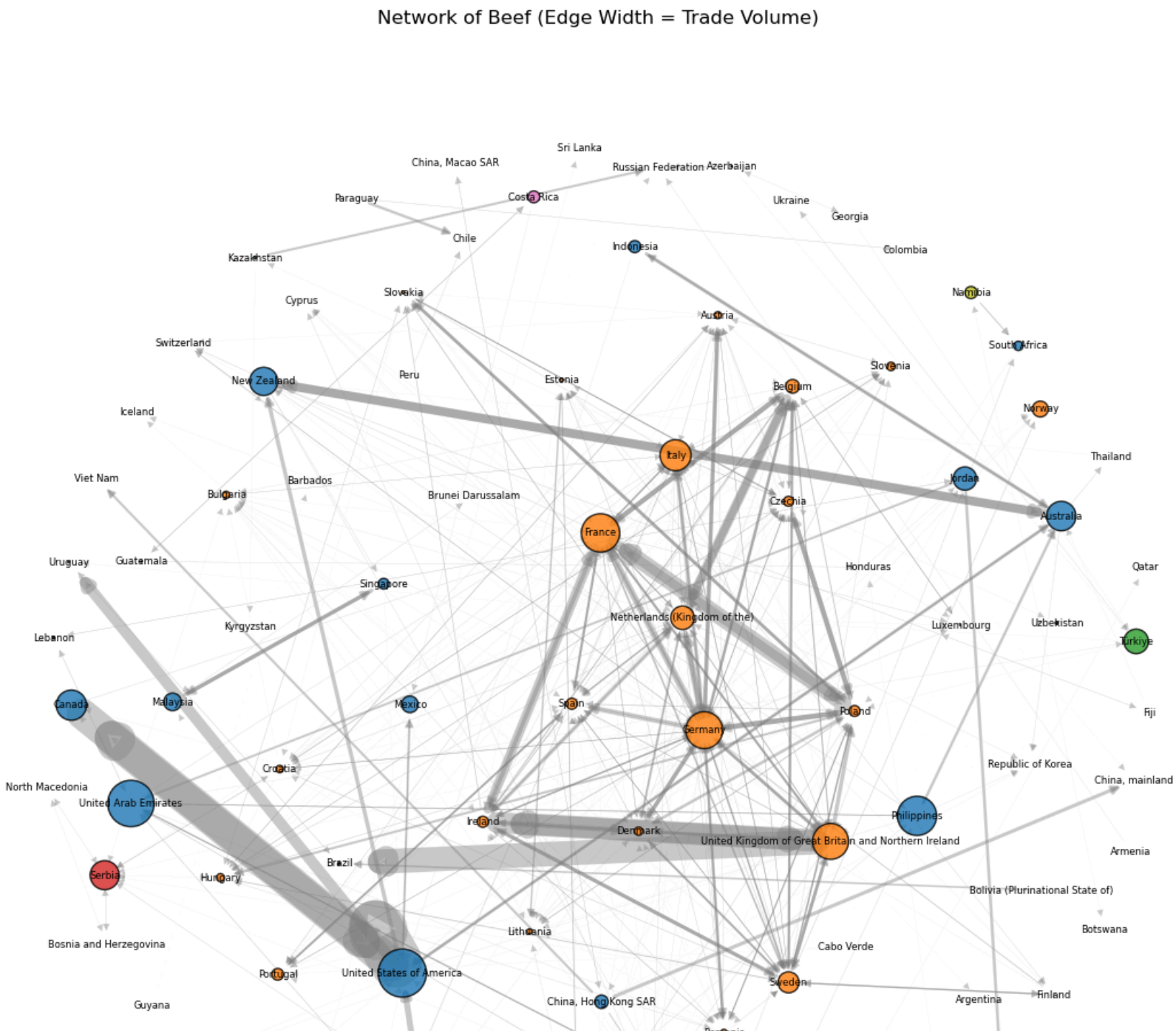
```
{'United States of America': 0, 'Mozambique': 0, 'Viet Nam': 0, 'Thailand': 0, 'China, mainland': 0, 'Canada': 0, 'Ghana': 0,
'Brunei Darussalam': 0, 'Singapore': 0, 'Djibouti': 0, 'Brazil': 0, 'Saudi Arabia': 0, 'Indonesia': 0, 'Oman': 0, 'New Zealan
d': 0, 'Bolivia (Plurinational State of)': 0, 'Uruguay': 0, 'Australia': 0, 'Kuwait': 0, 'Bahrain': 0, 'Egypt': 0, 'Republic of
Korea': 0, 'South Africa': 0, 'Malaysia': 0, 'Lebanon': 0, 'China, Hong Kong SAR': 0, 'Philippines': 0, 'Japan': 0, 'Qatar': 0,
'China, Macao SAR': 0, 'Mexico': 0, 'Fiji': 0, 'United Arab Emirates': 0, 'Jordan': 0, 'Jamaica': 0, 'China, Taiwan Province o
f': 0, 'Bulgaria': 1, 'Latvia': 1, 'Poland': 1, 'Finland': 1, 'Netherlands (Kingdom of the)': 1, 'Iceland': 1, 'France': 1, 'Ir
eland': 1, 'Switzerland': 1, 'Austria': 1, 'Germany': 1, 'Luxembourg': 1, 'Denmark': 1, 'Slovakia': 1, 'Lithuania': 1, 'Belgiu
m': 1, 'United Kingdom of Great Britain and Northern Ireland': 1, 'Morocco': 1, 'Czechia': 1, 'Slovenia': 1, 'Greece': 1, 'Croa
tia': 1, 'Romania': 1, 'Italy': 1, 'Estonia': 1, 'Cabo Verde': 1, 'Sweden': 1, 'Spain': 1, 'Norway': 1, 'Malta': 1, 'Hungary':
1, 'Portugal': 1, 'Argentina': 1, 'Cyprus': 1, 'Russian Federation': 2, 'Uzbekistan': 2, 'Türkiye': 2, 'Kyrgyzstan': 2, 'Kazakh
stan': 2, 'Azerbaijan': 2, 'Ukraine': 2, 'Armenia': 2, 'Georgia': 2, 'North Macedonia': 3, 'Montenegro': 3, 'Bosnia and Herzego
vina': 3, 'Serbia': 3, 'Peru': 4, 'Chile': 4, 'Paraguay': 4, 'Colombia': 4, 'El Salvador': 5, 'Costa Rica': 5, 'Honduras': 5,
'Guatemala': 5, 'Guyana': 6, 'Barbados': 6, 'Botswana': 7, 'Namibia': 7, 'Maldives': 8, 'Sri Lanka': 8}
```

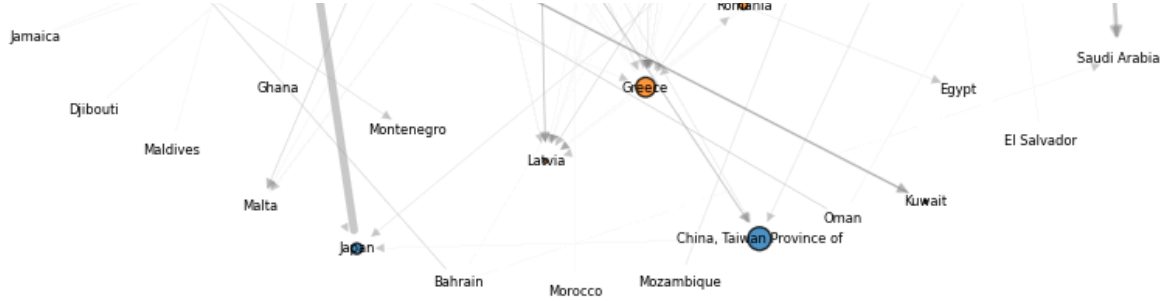
In [752…
```
nx.set_node_attributes(G_b, modularity_dict, 'modularity')
```

In [758…

```
# Extract node attributes
node_sizes = [8000 * G_b.nodes[node]['betweenness'] for node in G_b.nodes()]  # Scale betweenness
node_colors = [G_b.nodes[node]['modularity'] for node in G_b.nodes()]  # Modularity-based color

# Define colormap for modularity
cmap = plt.cm.get_cmap('tab10', max(node_colors) + 1)  # Adjust based on modularity classes

# Edge weight (두께): Final Value 기준으로 조정
edge_weights = [G_b[u][v]['Final Value'] * 0.0008 for u, v in G_b.edges()]  # 값 작으면 0.01로 올려도 됨

# Draw the network
plt.figure(figsize=(15, 15))
pos = nx.arf_layout(G_b, pos=None, scaling=1, a=1.1, etol=1e-06, dt=0.001, max_iter=1000)

# Draw nodes with size and color
nx.draw_networkx_nodes(G_b, pos,
                       node_size=node_sizes,
                       node_color=node_colors,
                       cmap=cmap,
                       alpha=0.8,
                       edgecolors='black')

# Draw edges with custom width
nx.draw_networkx_edges(G_b, pos,
                       width=edge_weights,
                       alpha=0.4,
                       edge_color='gray')

# Draw labels
nx.draw_networkx_labels(G_b, pos, font_size=6)

# Show the plot
plt.axis("off")
plt.title("Network of Beef (Edge Width = Trade Volume)")
plt.show()
plt.savefig('arf_of_Beef.png')
```

```
C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\639891186.py:6: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``mat
plotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```


Network of Beef (Edge Width = Trade Volume)
Sri Lanka
China, Macao SAR
Russian Federation
Azerbaijan
Paraguay
Costa Rica
Ukraine
Georgia
Chile
Indonesia
Colombia
Kazakhstan
Slovakia
Namibia
Cyprus
Austria
Switzerland
South Africa
Slovenia
New Zealand
Peru
Estonia
Belgium
Iceland
Norway
Italy
Thailand
Viet Nam
Barbados
Jordan
Bulgaria
Brunei Darussalam
Czechia
Australia
France
Uruguay
Guatemala
Honduras
Qatar
Singapore
Netherlands (Kingdom of the)
Luxembourg
Uzbekistan
Kyrgyzstan
Lebanon
Türkiye
Canada
Malaysia
Mexico
Spain
Poland
Fiji
Germany
Croatia
Republic of Korea
North Macedonia
China, mainland
United Arab Emirates
Philippines
Ireland
Denmark
United Kingdom of Great Britain and Northern Ireland
Armenia
Brazil
Serbia
Hungary
Bolivia (Plurinational State of)
Botswana
Lithuania
Bosnia and Herzegovina
Cabo Verde
Portugal
United States of America
Sweden
Finland
Guyana
China, Hong Kong SAR
Argentina



<Figure size 640x480 with 0 Axes>

In [760…
```
betweenness_dict = nx.betweenness_centrality(G) # Run betweenness centrality
eigenvector_dict = nx.eigenvector_centrality(G) # Run eigenvector centrality: How influential a node is in a network.

# Assign each to an attribute in your network
nx.set_node_attributes(G, betweenness_dict, 'betweenness')
nx.set_node_attributes(G, eigenvector_dict, 'eigenvector')
```

In [762…
```
communities = community.greedy_modularity_communities(G)

modularity_dict = {} # Create a blank dictionary
for i,c in enumerate(communities): # Loop through the list of communities
    for name in c: # Loop through each person in a community
        modularity_dict[name] = i # Create an entry in the dictionary for the person

print(modularity_dict)
```

```
{'India': 0, 'United States of America': 0, 'Costa Rica': 0, 'Viet Nam': 0, 'Ethiopia': 0, 'Thailand': 0, 'Brunei Darussalam':
0, 'Ghana': 0, 'Singapore': 0, 'China, mainland': 0, 'Canada': 0, 'Djibouti': 0, 'Brazil': 0, 'El Salvador': 0, 'Ecuador': 0,
'Indonesia': 0, 'Guatemala': 0, 'Sri Lanka': 0, 'Algeria': 0, 'Nepal': 0, 'New Zealand': 0, 'Bolivia (Plurinational State of)':
0, 'Argentina': 0, 'Uruguay': 0, 'Colombia': 0, 'Suriname': 0, 'United Republic of Tanzania': 0, 'Australia': 0, 'Togo': 0, 'Ca
mbodia': 0, 'Paraguay': 0, 'Maldives': 0, 'Republic of Korea': 0, 'Peru': 0, 'Iran (Islamic Republic of)': 0, 'Malaysia': 0, 'G
uyana': 0, 'China, Hong Kong SAR': 0, 'Honduras': 0, 'Philippines': 0, 'Japan': 0, 'China, Macao SAR': 0, 'Fiji': 0, 'Mexico':
0, 'Chile': 0, 'Jamaica': 0, 'China, Taiwan Province of': 0, 'Barbados': 0, 'Bulgaria': 1, 'Latvia': 1, 'Poland': 1, 'Finland':
1, 'Netherlands (Kingdom of the)': 1, 'Iceland': 1, 'France': 1, 'North Macedonia': 1, 'Uganda': 1, 'Switzerland': 1, 'Mauritiu
s': 1, 'Austria': 1, 'Germany': 1, 'Senegal': 1, 'Lithuania': 1, 'Luxembourg': 1, 'Denmark': 1, 'Slovakia': 1, 'Belgium': 1, 'U
nited Kingdom of Great Britain and Northern Ireland': 1, 'Bosnia and Herzegovina': 1, 'Cyprus': 1, 'Morocco': 1, 'Czechia': 1,
'Slovenia': 1, 'Montenegro': 1, 'Serbia': 1, 'Greece': 1, 'Romania': 1, 'Italy': 1, 'Estonia': 1, 'Croatia': 1, 'Cabo Verde':
1, 'Ukraine': 1, 'Sweden': 1, 'Spain': 1, 'Norway': 1, 'Malta': 1, 'Tunisia': 1, 'Hungary': 1, 'Portugal': 1, 'Ireland': 1, 'Ru
ssian Federation': 2, 'Turkmenistan': 2, 'Türkiye': 2, 'Yemen': 2, 'Kuwait': 2, 'Bahrain': 2, 'Egypt': 2, 'Kyrgyzstan': 2, 'Aze
rbaijan': 2, 'Saudi Arabia': 2, 'Armenia': 2, 'Uzbekistan': 2, 'Lebanon': 2, 'Oman': 2, 'Cameroon': 2, 'Republic of Moldova':
2, 'Kazakhstan': 2, 'Tajikistan': 2, 'Qatar': 2, 'United Arab Emirates': 2, 'Jordan': 2, 'Georgia': 2, 'Botswana': 3, 'Namibi
a': 3, 'Mozambique': 3, 'Malawi': 3, 'Eswatini': 3, 'South Africa': 3, 'Zambia': 3, 'Zimbabwe': 3, 'Burkina Faso': 4, "Côte d'I
voire": 4}
```

In [764…

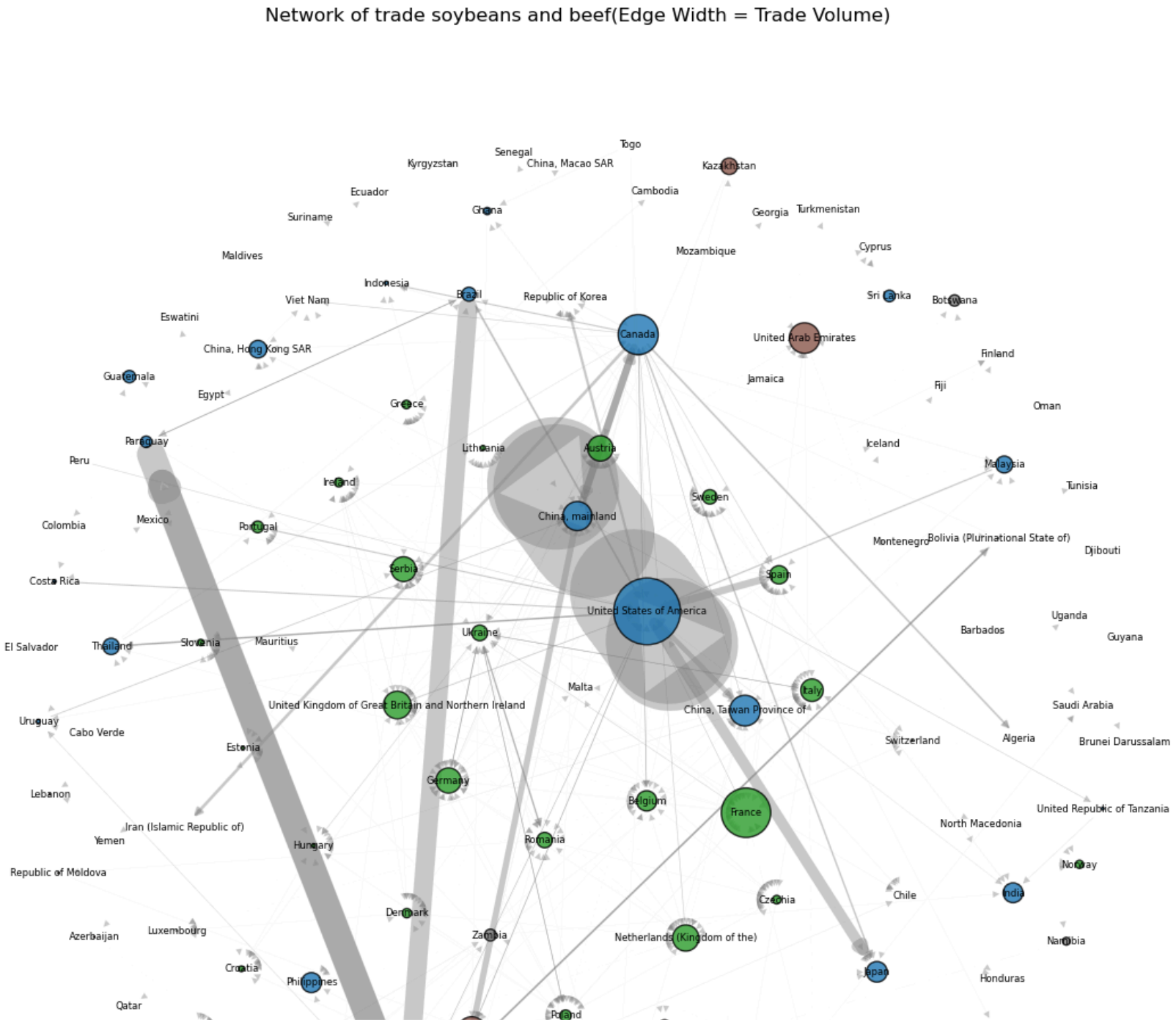
```python
nx.set_node_attributes(G, modularity_dict, 'modularity')
```

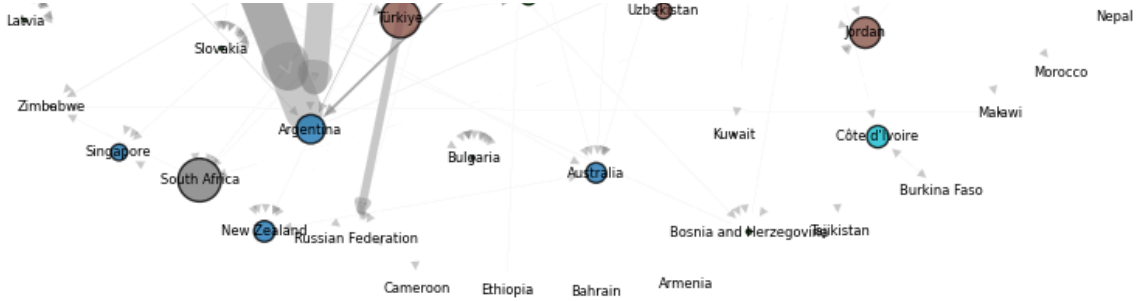
In [768…

```python
# Extract node attributes
node_sizes = [8000 * G.nodes[node]['betweenness'] for node in G.nodes()]  # Scale betweenness
node_colors = [G.nodes[node]['modularity'] for node in G.nodes()]  # Modularity-based color

# Define colormap for modularity
cmap = plt.cm.get_cmap('tab10', max(node_colors) + 1)  # Adjust based on modularity classes

# Edge weight
edge_weights = [G[u][v]['weight'] * 0.000003 for u, v in G.edges()]  # 값 작으면 0.01로 올려도 됨

# Draw the network
plt.figure(figsize=(15, 15))
pos = nx.arf_layout(G, pos=None, scaling=1, a=1.1, etol=1e-06, dt=0.001, max_iter=1000)

# Draw nodes with size and color
nx.draw_networkx_nodes(G, pos,
                       node_size=node_sizes,
                       node_color=node_colors,
                       cmap=cmap,
                       alpha=0.8,
```

```
                        edgecolors='black')

# Draw edges with custom width
nx.draw_networkx_edges(G, pos,
                        width=edge_weights,
                        alpha=0.4,
                        edge_color='gray')

# Draw labels
nx.draw_networkx_labels(G, pos, font_size=6)

# Show the plot
plt.axis("off")
plt.title("Network of trade soybeans and beef(Edge Width = Trade Volume)")
plt.show()
plt.savefig("TradeVolume of soybeans and beef.png")
```

```
C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\4083612349.py:6: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``mat
plotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```


Network of trade soybeans and beef(Edge Width = Trade Volume)
Togo
Senegal
Kyrgyzstan
China, Macao SAR
Kazakhstan
Ecuador
Cambodia
Ghana
Georgia
Turkmenistan
Suriname
Maldives
Mozambique
Cyprus
Indonesia
Brazil
Republic of Korea
Sri Lanka
Botswana
Viet Nam
Eswatini
Canada
United Arab Emirates
China, Hong Kong SAR
Finland
Guatemala
Jamaica
Fiji
Egypt
Greece
Oman
Paraguay
Lithuania
Austria
Iceland
Peru
Malaysia
Ireland
Tunisia
Sweden
China, mainland
Colombia
Mexico
Portugal
Montenegro
Bolivia (Plurinational State of)
Djibouti
Serbia
Spain
Costa Rica
United States of America
Uganda
Barbados
Guyana
El Salvador
Thailand
Slovenia
Mauritius
Ukraine
Malta
Italy
United Kingdom of Great Britain and Northern Ireland
China, Taiwan Province of
Saudi Arabia
Uruguay
Cabo Verde
Switzerland
Algeria
Brunei Darussalam
Estonia
Germany
Lebanon
Belgium
France
United Republic of Tanzania
Iran (Islamic Republic of)
North Macedonia
Yemen
Hungary
Romania
Norway
Republic of Moldova
Chile
India
Czechia
Denmark
Azerbaijan
Luxembourg
Zambia
Netherlands (Kingdom of the)
Namibia
Croatia
Philippines
Japan
Honduras
Qatar
Poland




```
<Figure size 640x480 with 0 Axes>
```

Flow visualization on map

In [370…

```
import plotly.graph_objects as go

df_airports = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2011_february_us_airport_traffic.csv')
print(df_airports.head())

df_flight_paths = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2011_february_aa_flight_paths.csv')
print(df_flight_paths.head())
```

```
  iata                          airport              city state country  \
0  ORD       Chicago O'Hare International           Chicago    IL     USA
1  ATL  William B Hartsfield-Atlanta Intl           Atlanta    GA     USA
2  DFW    Dallas-Fort Worth International  Dallas-Fort Worth    TX     USA
3  PHX   Phoenix Sky Harbor International           Phoenix    AZ     USA
4  DEN                       Denver Intl            Denver    CO     USA

         lat        long    cnt
0  41.979595  -87.904464  25129
1  33.640444  -84.426944  21925
2  32.895951  -97.037200  20662
3  33.434167 -112.008056  17290
4  39.858408 -104.667002  13781
   start_lat  start_lon    end_lat     end_lon airline airport1 airport2  cnt
0  32.895951 -97.037200  35.040222 -106.609194      AA      DFW      ABQ  444
1  41.979595 -87.904464  30.194533  -97.669872      AA      ORD      AUS  166
2  32.895951 -97.037200  41.938874  -72.683228      AA      DFW      BDL  162
3  18.439417 -66.001833  41.938874  -72.683228      AA      SJU      BDL   56
4  32.895951 -97.037200  33.562943  -86.753550      AA      DFW      BHM  168
```

In [373…

```
fig = go.Figure()

fig.add_trace(go.Scattergeo(
    locationmode = 'USA-states',
    lon = df_airports['long'],
    lat = df_airports['lat'],
    hoverinfo = 'text',
    text = df_airports['airport'],
    mode = 'markers',
    marker = dict(
        size = 2,
        color = 'rgb(255, 0, 0)',
        line = dict(
            width = 3,
            color = 'rgba(68, 68, 68, 0)'
        )
    )))

flight_paths = []
for i in range(len(df_flight_paths)):
    fig.add_trace(
```

```
        go.Scattergeo(
            locationmode = 'USA-states',
            lon = [df_flight_paths['start_lon'][i], df_flight_paths['end_lon'][i]],
            lat = [df_flight_paths['start_lat'][i], df_flight_paths['end_lat'][i]],
            mode = 'lines',
            line = dict(width = 1,color = 'red'),
            opacity = float(df_flight_paths['cnt'][i]) / float(df_flight_paths['cnt'].max()),
        )
    )

fig.update_layout(
    title_text = 'Feb. 2011 American Airline flight paths<br>(Hover for airport names)',
    showlegend = False,
    geo = dict(
        scope = 'north america',
        projection_type = 'azimuthal equal area',
        showland = True,
        landcolor = 'rgb(243, 243, 243)',
        countrycolor = 'rgb(204, 204, 204)',
    ),
)

fig.show()
```

In [374…

```
import geopandas as gpd

# (1) Read shp file --- polygon file
gdf = gpd.read_file("world_countries_new.shp")

# (2) Get centroids of countries
gdf['centroid'] = gdf.geometry.centroid
```

```
# (3) Get lon and lat
gdf['lon'] = gdf.centroid.x
gdf['lat'] = gdf.centroid.y

# (4) Select columns
gdf = gdf[['Area','Area_Cd','lon', 'lat']].copy()
gdf = gdf.dropna(subset=['Area_Cd'])


gdf
```

```
C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\3319270481.py:7: UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometrie
s to a projected CRS before this operation.


C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\3319270481.py:10: UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometrie
s to a projected CRS before this operation.


C:\Users\JuhyunN\AppData\Local\Temp\ipykernel_21112\3319270481.py:11: UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometrie
s to a projected CRS before this operation.
```

Out[374…

	Area	Area_Cd	lon	lat
1	Guyana	91.0	-58.974781	4.792017
4	Suriname	207.0	-55.911827	4.126394
5	Trinidad and Tobago	220.0	-61.253172	10.468645
6	Venezuela (Bolivarian Republic of)	236.0	-66.169564	7.122449
11	Samoa	244.0	-172.159459	-13.758367
...	...	...	...	...
223	China, Taiwan Province of	214.0	120.968186	23.747486
224	China, Hong Kong SAR	96.0	114.134965	22.381121
225	China, Macao SAR	128.0	113.559255	22.160367
226	Sudan	276.0	29.953612	15.994194
227	China, mainland	41.0	103.809684	36.621082

200 rows × 4 columns

Convert the node info of network to a dataframe

In [610…

```
in_degree_soy_dict = dict(G_s.in_degree(weight='weight'))
nx.set_node_attributes(G_s, in_degree_soy_dict, 'in_degree_soy')

soybean_data = pd.DataFrame.from_dict(dict(G_s.nodes(data=True)), orient='index')

# Have node IDs as a column
soybean_data.reset_index(inplace=True)
soybean_data.rename(columns={'index': 'node'}, inplace=True)
soybean_data
```

Out[610…

	node	betweenness_soybean	eigenvector_soybean	in_degree_soy
0	Togo	0.000000	2.512281e-17	0
1	Belgium	0.035509	2.407475e-01	17
2	Canada	0.107372	1.055907e-01	10
3	Algeria	0.000000	8.376930e-03	1
4	Luxembourg	0.000000	6.326154e-02	4
...	...	...	...	...
92	Morocco	0.000000	2.365505e-02	1
93	Tunisia	0.000000	1.149093e-02	1
94	Malta	0.000000	1.601795e-02	1
95	Senegal	0.000000	2.365505e-02	1
96	Uganda	0.000000	1.601795e-02	1

97 rows × 4 columns

In [695…

```
in_degree_beef_dict = dict(G_b.in_degree(weight='weight'))
nx.set_node_attributes(G_b, in_degree_beef_dict, 'in_degree_beef')

beef_data = pd.DataFrame.from_dict(dict(G_b.nodes(data=True)), orient='index')

# Have node IDs as a column
beef_data.reset_index(inplace=True)
beef_data.rename(columns={'index': 'node'}, inplace=True)
beef_data
```

Out[695...

	node	in_degree_beef
0	Fiji	1
1	Australia	7
2	Philippines	7
3	Guyana	0
4	Barbados	1
...	...	...
92	Singapore	2
93	Thailand	1
94	Viet Nam	1
95	Ukraine	1
96	Malta	3

97 rows × 2 columns

In [689...

```
soybean_data['node'] = soybean_data['node'].astype(str)
gdf['Area'] = gdf['Area'].astype(str)

merged_node_s = pd.merge(
    soybean_data,
    gdf[['Area', 'lon', 'lat']],
    left_on='node',
    right_on='Area',
    how='left'
)

merged_node_s
```

Out[689…

	node	betweenness_soybean	eigenvector_soybean	in_degree_soy	Area	lon	lat
0	Togo	0.000000	2.512281e-17	0	Togo	0.975721	8.534961
1	Belgium	0.035509	2.407475e-01	17	Belgium	4.663987	50.642852
2	Canada	0.107372	1.055907e-01	10	Canada	-98.265331	61.392017
3	Algeria	0.000000	8.376930e-03	1	Algeria	2.632388	28.163240
4	Luxembourg	0.000000	6.326154e-02	4	Luxembourg	6.087814	49.770629
...	...	...	...	...	...	...	...
92	Morocco	0.000000	2.365505e-02	1	Morocco	-6.317846	31.883625
93	Tunisia	0.000000	1.149093e-02	1	Tunisia	9.561336	34.110860
94	Malta	0.000000	1.601795e-02	1	Malta	14.441922	35.890523
95	Senegal	0.000000	2.365505e-02	1	Senegal	-14.467653	14.366965
96	Uganda	0.000000	1.601795e-02	1	Uganda	32.386218	1.279964

97 rows × 7 columns

In [697…

```
beef_data['node'] = beef_data['node'].astype(str)
gdf['Area'] = gdf['Area'].astype(str)

merged_node_b = pd.merge(
    beef_data,
    gdf[['Area', 'lon', 'lat']],
    left_on='node',
    right_on='Area',
    how='left'
)

merged_node_b
```

Out[697…

	node	in_degree_beef	Area	lon	lat
0	Fiji	1	Fiji	171.983186	-17.453531
1	Australia	7	Australia	134.489563	-25.734969
2	Philippines	7	Philippines	122.878708	11.741833
3	Guyana	0	Guyana	-58.974781	4.792017
4	Barbados	1	Barbados	-59.561955	13.178714
...	...	...	...	...	...
92	Singapore	2	Singapore	103.808051	1.351616
93	Thailand	1	Thailand	101.017361	15.127036
94	Viet Nam	1	Viet Nam	106.301474	16.659257
95	Ukraine	1	Ukraine	31.387115	49.017088
96	Malta	3	Malta	14.441922	35.890523

97 rows × 5 columns

In [699…

```
edge_data_s = pd.DataFrame([
    {'source': u,'target': v, **data}
    for u, v, data in G_s.edges(data=True)
])
edge_data_s
```

Out[699…

	source	target	Final Value
0	Togo	Belgium	9298.05
1	Togo	China, mainland	965.42
2	Togo	Ghana	20000.00
3	Togo	United States of America	11888.70
4	Belgium	Austria	1540.70
...	...	...	...
464	Portugal	United States of America	263832.11
465	Indonesia	Japan	1.00
466	Indonesia	Malaysia	6331.68
467	Ireland	Netherlands (Kingdom of the)	706.23
468	Ireland	United Kingdom of Great Britain and Northern I...	3095.39

469 rows × 3 columns

In [701…

```
edge_data_b = pd.DataFrame([
    {'source': u,'target': v, **data}
    for u, v, data in G_b.edges(data=True)
])
edge_data_b
```

Out[701…

	source	target	Final Value
0	Fiji	Australia	63.69
1	Fiji	New Zealand	154.12
2	Australia	Indonesia	1741.89
3	Australia	New Zealand	6657.30
4	Australia	Philippines	93.55
...	...	...	...
549	Switzerland	United Kingdom of Great Britain and Northern I...	19.21
550	Singapore	Malaysia	2974.69
551	Singapore	Republic of Korea	1.08
552	Singapore	United States of America	44.19
553	Thailand	New Zealand	8.96

554 rows × 3 columns

In [702…

```
merged_edge_s = pd.merge(
    edge_data_s, gdf,
    left_on=['source'],
    right_on=['Area'],
    how='inner'  # or 'left', 'right', 'outer'
)

merged_edge_s = merged_edge_s.drop(columns=['Area', 'Area_Cd'])
merged_edge_s = merged_edge_s.rename(columns={
    'lon': 'start_lon',
    'lat': 'start_lat'
})
merged_edge_s
```

Out[702...

	source	target	Final Value	start_lon	start_lat
0	Togo	Belgium	9298.05	0.975721	8.534961
1	Togo	China, mainland	965.42	0.975721	8.534961
2	Togo	Ghana	20000.00	0.975721	8.534961
3	Togo	United States of America	11888.70	0.975721	8.534961
4	Belgium	Austria	1540.70	4.663987	50.642852
...	...	...	...	...	...
415	Portugal	United States of America	263832.11	-8.562741	39.600988
416	Indonesia	Japan	1.00	117.300426	-2.230214
417	Indonesia	Malaysia	6331.68	117.300426	-2.230214
418	Ireland	Netherlands (Kingdom of the)	706.23	-8.150581	53.176380
419	Ireland	United Kingdom of Great Britain and Northern I...	3095.39	-8.150581	53.176380

420 rows × 5 columns

In [705...

```
merged_edge_b = pd.merge(
    edge_data_b, gdf,
    left_on=['source'],
    right_on=['Area'],
    how='inner'  # or 'left', 'right', 'outer'
)

merged_edge_b = merged_edge_b.drop(columns=['Area', 'Area_Cd'])
merged_edge_b = merged_edge_b.rename(columns={
    'lon': 'start_lon',
    'lat': 'start_lat'
})
merged_edge_b
```

Out[705…

	source	target	Final Value	start_lon	start_lat
0	Fiji	Australia	63.69	171.983186	-17.453531
1	Fiji	New Zealand	154.12	171.983186	-17.453531
2	Australia	Indonesia	1741.89	134.489563	-25.734969
3	Australia	New Zealand	6657.30	134.489563	-25.734969
4	Australia	Philippines	93.55	134.489563	-25.734969
...	...	...	...	...	...
492	Switzerland	United Kingdom of Great Britain and Northern I...	19.21	8.234393	46.802496
493	Singapore	Malaysia	2974.69	103.808051	1.351616
494	Singapore	Republic of Korea	1.08	103.808051	1.351616
495	Singapore	United States of America	44.19	103.808051	1.351616
496	Thailand	New Zealand	8.96	101.017361	15.127036

497 rows × 5 columns

Merge 'target' info with the centroid

In [708…

```
merged_edge_s = pd.merge(
    merged_edge_s, gdf,
    left_on=['target'],
    right_on=['Area'],
    how='inner'  # or 'left', 'right', 'outer'
)

merged_edge_s = merged_edge_s.drop(columns=['Area', 'Area_Cd'])
merged_edge_s = merged_edge_s.rename(columns={
    'lon': 'end_lon',
    'lat': 'end_lat'
```

```
})
merged_edge_s
```

Out[708...

	source	target	Final Value	start_lon	start_lat	end_lon	end_lat
0	Togo	Belgium	9298.05	0.975721	8.534961	4.663987	50.642852
1	Togo	China, mainland	965.42	0.975721	8.534961	103.809684	36.621082
2	Togo	Ghana	20000.00	0.975721	8.534961	-1.207301	7.959848
3	Togo	United States of America	11888.70	0.975721	8.534961	-112.491510	45.695577
4	Belgium	Austria	1540.70	4.663987	50.642852	14.140191	47.592903
...	...	...	...	...	...	...	...
393	Portugal	Germany	27.04	-8.562741	39.600988	10.393646	51.106562
394	Portugal	Spain	22820.89	-8.562741	39.600988	-3.649566	40.226829
395	Portugal	United States of America	263832.11	-8.562741	39.600988	-112.491510	45.695577
396	Indonesia	Japan	1.00	117.300426	-2.230214	137.990744	37.562161
397	Indonesia	Malaysia	6331.68	117.300426	-2.230214	109.708192	3.792368

398 rows × 7 columns

In [710…

```
merged_edge_b = pd.merge(
    merged_edge_b, gdf,
    left_on=['target'],
    right_on=['Area'],
    how='inner'  # or 'left', 'right', 'outer'
)

merged_edge_b = merged_edge_b.drop(columns=['Area', 'Area_Cd'])
merged_edge_b = merged_edge_b.rename(columns={
    'lon': 'end_lon',
    'lat': 'end_lat'
})
merged_edge_b
```

Out[710…

	source	target	Final Value	start_lon	start_lat	end_lon	end_lat
0	Fiji	Australia	63.69	171.983186	-17.453531	134.489563	-25.734969
1	Fiji	New Zealand	154.12	171.983186	-17.453531	171.779900	-41.838874
2	Australia	Indonesia	1741.89	134.489563	-25.734969	117.300426	-2.230214
3	Australia	New Zealand	6657.30	134.489563	-25.734969	171.779900	-41.838874
4	Australia	Philippines	93.55	134.489563	-25.734969	122.878708	11.741833
...	...	...	...	...	...	...	...
449	Switzerland	Germany	249.37	8.234393	46.802496	10.393646	51.106562
450	Singapore	Malaysia	2974.69	103.808051	1.351616	109.708192	3.792368
451	Singapore	Republic of Korea	1.08	103.808051	1.351616	127.834773	36.375090
452	Singapore	United States of America	44.19	103.808051	1.351616	-112.491510	45.695577
453	Thailand	New Zealand	8.96	101.017361	15.127036	171.779900	-41.838874

454 rows × 7 columns

Visualization

Subset the link based on the value

In [770…

```
fig = go.Figure()

fig.add_trace(go.Scattergeo(
    #locationmode = 'USA-states',
    lon = merged_node_s['lon'],
    lat = merged_node_s['lat'],
    hoverinfo = 'text',
    text = merged_node_s['node'],
    mode = 'markers',
    marker = dict(
        size = merged_node_s['in_degree_soy'],
        sizemode = 'area',              # or 'diameter'
        sizeref = 2.*max(merged_node_s['in_degree_soy'])/(40.**2),  # scale sizes
        sizemin = 2,                    # minimum size of dots
        color = 'rgb(255, 0, 0)',
        line = dict(
            width = 3,
            color = 'rgba(68, 68, 68, 0)'
        )
    )))

flight_paths = []
for i in range(len(merged_edge_s)):
    value = merged_edge_s['Final Value'][i]
    max_value = merged_edge_s['Final Value'].max()


    fig.add_trace(
        go.Scattergeo(
            #locationmode = 'USA-states',
            lon = [merged_edge_s['start_lon'][i], merged_edge_s['end_lon'][i]],
            lat = [merged_edge_s['start_lat'][i], merged_edge_s['end_lat'][i]],
            mode = 'lines',
            line = dict(width = 0.5 + 4 * (value / max_value),
```

```
                        color = 'red'),
            opacity = 0.6,
        )
    )

fig.update_layout(
    title_text = '2023 Global Soybean trade<br>(Hover for country names)',
    showlegend = False,
    geo = dict(
        scope = 'world',
        projection_type = 'equirectangular',
        showland = True,
        landcolor = 'rgb(243, 243, 243)',
        countrycolor = 'rgb(204, 204, 204)',
    ),
)

fig.show()
fig.write_html("soybean_trade_map.html")
```

In [772…

```
fig = go.Figure()

fig.add_trace(go.Scattergeo(
    #locationmode = 'USA-states',
    lon = merged_node_b['lon'],
    lat = merged_node_b['lat'],
    hoverinfo = 'text',
    text = merged_node_b['node'],
```

```
    mode = 'markers',
    marker = dict(
        size = merged_node_b['in_degree_beef'],
        sizemode = 'area',              # or 'diameter'
        sizeref = 2.*max(merged_node_b['in_degree_beef'])/(40.**2),  # scale sizes
        sizemin = 2,                    # minimum size of dots
        color = 'rgb(255, 0, 0)',
        line = dict(
            width = 3,
            color = 'rgba(68, 68, 68, 0)'
        )
    )))

flight_paths = []
for i in range(len(merged_edge_b)):
    value = merged_edge_b['Final Value'][i]
    max_value = merged_edge_b['Final Value'].max()


    fig.add_trace(
        go.Scattergeo(
            #locationmode = 'USA-states',
            lon = [merged_edge_b['start_lon'][i], merged_edge_b['end_lon'][i]],
            lat = [merged_edge_b['start_lat'][i], merged_edge_b['end_lat'][i]],
            mode = 'lines',
            line = dict(width = 0.5 + 4 * (value / max_value),
                        color = 'red'),
            opacity = 0.6,
        )
    )

fig.update_layout(
    title_text = '2023 Global Beef trade<br>(Hover for country names)',
    showlegend = False,
    geo = dict(
        scope = 'world',
        projection_type = 'equirectangular',
        showland = True,
        landcolor = 'rgb(243, 243, 243)',
        countrycolor = 'rgb(204, 204, 204)',
    ),
```

```
)

fig.show()
fig.write_html("beef_trade_map.html")
```

In [ ]: