Oxford Internet Institute, University of Oxford

Assignment Cover Sheet

Candidate Number	1029100
Assignment	Statistical analysis of networks
Term	Hilary Term 2019
Title/Question	Motifs in Global Migration Networks, 1990-2017
Word Count	2448

By placing a tick in this box ☑ I hereby certify as follows:

(a) This thesis or coursework is entirely my own work, except where acknowledgments of other sources are given. I also confirm that this coursework has not been submitted, wholly or substantially, to another examination at this or any other University or educational institution;

(b) I have read and understood the Education Committee's information and guidance on academic good practice and plagiarism at `https://www.ox.ac.uk/students/academic/guidance/skills?wssl=1`.

(c) I agree that my work may be checked for plagiarism using Turnitin software and have read the Notice to Candidates which can be seen at: `http://www.admin.ox.ac.uk/proctors/turnitin2w.shtml`, and that I agree to my work being screened and used as explained in that Notice;

(d) I have clearly indicated (with appropriate references) the presence of all material I have paraphrased, quoted or used from other sources, including any diagrams, charts, tables or graphs.

(e) I have acknowledged appropriately any assistance I have received in addition to that provided by my [tutor/supervisor/adviser].

(f) I have not sought assistance from a professional agency;

(g) I understand that any false claims for this work will be reported to the Proctors and may be penalized in accordance with the University regulations.

Please remember:

- To attach a second relevant cover sheet if you have a disability such as dyslexia or dyspraxia. These are available from the Higher Degrees Office, but the Disability Advisory Service will be able to guide you.

Motifs in Global Migration Networks, 1990-2017

1029100

1 Introduction

In recent years, migration has emerged as one of the most controversial topics in policy circles. It remains to be a consistent theme amongst voters and politicians who are concerned with its impact on national security, labour and employment, and overall economic welfare.

Critics note that migrants are mostly from developing countries, and that their attraction to the promise of higher wages and a better quality of life comes at the expense of citizens whose jobs and economic resources are "being taken away"(Hoban, 2017). Another observation is that immigrants hail from conflict-ridden, war-torn, and fragile nations which may pose security risks to the destination country (Koser, 2001). My goal with this paper is to use statistical analysis to investigate whether these two themes emerge in migration networks. *Is migration to the developed world indeed driven by citizens from developing countries and conflict states?*

By analyzing network motifs – or overrepresented small graphs in the immigration network, I observe patterns in the topology and structure of the network, which could then provide information on the behaviour and flow of migrants globally. I particularly focus on one class of motifs: the *triad*, or ordered triples that are generally mapped to 16 types.

Comparing the immigration network to a directed and unweighted configuration model, I find that three motifs emerge (*210, 120D, and 300*, explained in the next section on definitions). An analysis of the countries in these motifs suggest that immigration to high-income economies remains elusive to lower-income countries. If anything, high-income economies enjoy the mobility to migrate freely to other countries, while lower and middle-income economies do not appear to have the same benefit. There is also very little evidence that points to substantial migration from fragile states, though it seems that the weighted network may provide clues on an emerging trend. Thus, I suggest that in a future analysis, one may explore the possibility of weighted motifs in order to better understand the dynamics of migration flows.

2 Definitions

The concept of network motifs were first introduced by Milo et al. (2002), and described them as "patterns of interconnections occurring in complex networks at numbers that are significantly higher than those in randomized networks". Simply put, network motifs look at recurring sub-structures in a large graph, and offers a richer understanding of the underlying mechanisms in complex networks that may be overlooked by global network statistics.

The general idea of motifs is to search for 3- or 4-node subgraphs in a directed, unipartite network. Among the many real-world applications of motifs include: the identification of protein interactions in biology, food webs in ecology, and dynamic processes in electronic circuits (Milo et al., 2002). In economic systems, it has helped provide early warning signals of risk and financial collapse (Squartini et al., 2013).

2.1 Triad census

For purposes of this analysis we limit ourselves to the detection of triads, or 3-node subgraphs. Given that each of the three nodes can connect to two others, there are 6 possible ties which could either be present or absent. Mathematically, there are $2^6 = 64$ triad states, or possible values (Wasserman & Faust, 1994). Some of these states are *isomorphic*, or redundant: by removing the node labels and retaining only edge directions, these states are equivalent. Hence, the final triad census is narrowed to 16 isomorphism types, shown in Figure 1.

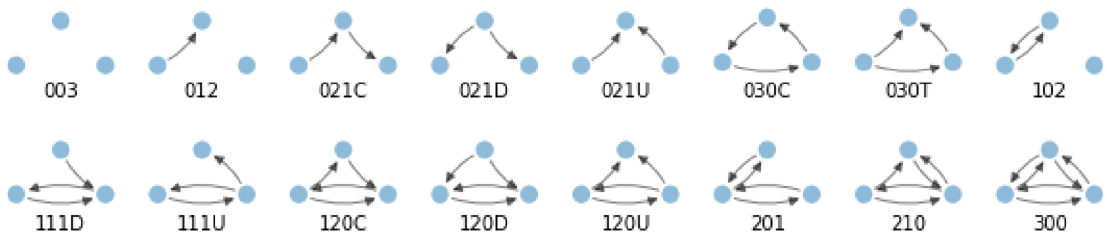


Figure 1: The triad census of 16 isomorphism classes. The labeling convention for each triad is described in appendix A.

2.2 Migrant stock

I use the United Nations' definition for international migrants, equating to a country's foreign-born population and/or citizens with a different country of citizenship. The migrant stock thus includes refugees, and excludes naturalized persons.

3 Methods

3.1 Dataset

The data is taken from the United Nations database (2017), which contains a weighted adjacency matrix of international migrants by origin for the every 5 years from 1990 to 2017. For this analysis, I compare the years 1990, 2000, 2010, and 2017. Per the UN documentation, data is available for all countries and areas of the world. The dataset also contains metadata for each country, such as geographic region and income classification (low, middle, high).

3.2 Network statistics

I introduce the following terminology that provides descriptive and summary information on the structure of the immigration network.

- **In-degree:** refers to nodes that are adjacent *to* another node. In migration parlance, I define in-degrees as the destination countries by other countries.
- **Out degree:** refers to nodes that are adjacent *from* another node. Country leavers are considered out-degrees.
- **Weighted in/out-degree:** is similar to the definitions above, instead that the edges are equal to the volume of people arriving to their destination country (in the case of in-degree) or leaving their country of origin (in the case of out-degree).
- **Betweenness centrality:** implies node importance, i.e., the frequency that a country appears in between the shortest paths of many other countries.
- **Transitivity or global clustering coefficient:** captures the notion of global integration in migrant flows: if country a is connected to countries b and c, transitivity measures the likelihood that country b and c are also connected.
- **Average path length:** is a measure of reachability between two nodes. In migration, this implies the shortest route that one country can migrate to another and vice versa.

3.3 Motif detection

For the years 1990, 2000, 2010, and 2017, I perform a triad census as enumerated in Figure 1, or a tally of the number of occurrences of each triad in the graph.

3.4 Monte Carlo test

The next step is to confirm whether the triad census types appear significantly larger (a motif) or smaller (an anti-motif) to a baseline graph. Hence, I compare my results to two models, a directed configuration graph and an Erdos-Renyi directed random graph.

For the configuration model, I replicated the in- and out-degree distributions of the migration graph as inputs to the model, and performed a triad census on 49 iterations of the configuration graph. After which, I compared the distribution of the census counts generated from the model graph with the census counts from the migration graph. Statistical significance of deviations to the model distribution was determined via the *significance profile*, explained in the next subsection.

Meanwhile, for the Erdos-Renyi random graph model, I used the nodes and directed edges of the migration graph as inputs, and similarly performed a triad census on 49 iterations of the random graph.

3.5 Significance profile

To determine whether a triad is significantly different from the model configuration (i.e., a motif), I perform a standard z-test for each triad i:

$$Z_i = \frac{x_i - \mu_i}{\sigma_i}$$

where x is the census count for (migration) triad i, while μ and σ are the average, and standard deviations of the census count for (49 runs of the configuration model) triad i.

Since the results of the z-test comprise extreme value ranges for each triad, for comparability across triad census types, I limit the range from -1 to 1 via a normalization trick taken from Milo et al. (2004) called a *significance profile*:

$$SP_i = \frac{Z_i}{(\sum {Z_i}^2)^{1/2}}$$

By convention, significance profiles (SPs) show relative rather than absolute significance, which is important for motif analysis whose z-scores may be influenced by graph size (Milo et al., 2004). Thus, I arbitrarily rank SP values to determine which motifs and/or anti-motifs are most prominent.

4 Analysis

4.1 Summary statistics

Figure 2 shows a visualization of the global immigration network using the latest data from the United Nations. Visually we can observe a highly-connected network, which can be confirmed by the network statistics in Table 1. For each decade, we can see an increasing number of edge connections and higher in/out degrees, which signals higher graph density over time. Similarly, the shorter average path lengths may imply improved global integration from 1990-2017.

Figure 2: Directed, weighted immigration network of the world in 2017. Size of nodes indicate out-degrees, and edge weights denote migrant volume.

Interestingly, our analysis of migration trends may differ depending on our treatment of edge weights. For example when examining in-degrees, it appears that Chile is a popular destination globally – migrants are diverse, and hail from over 210 countries. However, when considering edge weights, Chile does not appear amongst the top; instead, the United States ranks consistently –and convincingly– as the top destination for migrants by volume. This may be due to the fact that the US attracts migrants of the same origin, which one can match from the list of weighted out-degrees: India and Mexico are the top migrants to the US (Migration Policy Institute, 2017). Notably, war-torn Syria is ranked 6th by weighted out-degree.

Meanwhile, five countries appear to be the most influential and central to migrant flows: France, UK, USA, Australia and Canada. This may be reflective of the country's liberal immigration policies, as some sources suggest (Barder & Krylova, 2016). On the other hand, the weighted betweenness centrality makes little sense– a tiny country, Liechetenstein appears at the top. Curiously, other small countries are also on the list (Estonia, Guinea, Cabo Verde, etc). Their centrality position may be indicative of the strength of their ties rather than themselves.

Thus, edge weights may provide different perspectives on the immigration network. Unweighted degrees may indicate migrant diversity, but in policy planning, perhaps the volume of migrants may be more crucial. Conversely, unweighted betweenness centrality may be more important in international policy, but in local policy planning, it may also be beneficial to know the strengths of your migratory alliances and the weight of their influence.

Table 1: Immigration network summary statistics

	1990	2000	2010	2017
Totals				
Nodes	232	232	232	232
Edges	10317	10559	11095	11137
Transitivity	0.600	0.609	0.616	0.614
Averages				
Degree (in/out)	44	46	48	48
Degree (weighted, in/out)	620,928	712,996	912,646	1,065,324
Betweenness centrality	0.004	0.004	0.004	0.004
Betweenness centrality (weighted)	0.017	0.015	0.014	0.015
Path length	1.928	1.922	1.909	1.905
Top five countries				
In-degrees	Australia: 211 Greece: 209 France: 206 UK: 203 Denmark: 196	Chile: 210 Greece: 209 France: 206 UK: 205 Ireland: 195	Chile: 210 France: 206 UK: 205 Australia: 204 Canada: 197	Chile: 210 Australia: 206 UK: 205 France: 205 Canada: 197
In-degrees (weighted)	USA: 20,134,790 Russia: 11,516,298 India: 7,362,652 Ukraine: 6,481,438 Pakistan: 6,203,799	USA: 33,157,941 Russia: 11,891,623 Germany: 8,658,910 India: 6,286,286 France: 6,278,718	USA: 42,071,829 Russia: 11,194,137 Germany: 9,711,410 Saudi: 8,147,064 UK: 7,560,559	USA: 47,412,413 Germany: 12,044,115 Saudi: 11,774,584 Russia: 11,650,842 UK: 8,799,334
Out-degrees	USA: 157 UK: 140 China: 138 France: 135 Canada: 123	USA: 158 UK: 141 China: 139 France: 136 India: 123	USA: 162 UK: 145 China: 143 France: 138 India: 129	USA: 162 UK: 146 China: 143 France: 138 India: 130
Out-degrees (weighted)	Russia: 12,664,537 Afghanistan: 6,724,681 India: 6,718,862 Ukraine: 5,549,477 Bangladesh: 5,451,546	Russia: 10,734,963 Mexico: 9,562,278 India: 7,978,365 China: 5,786,954 Ukraine: 5,596,463	India: 13,321,332 Mexico: 12,413,085 Russia: 10,213,313 China: 8,648,885 Bangladesh: 6,742,845	India: 16,587,720 Mexico: 12,964,882 Russia: 10,635,994 China: 9,962,058 Bangladesh: 7,499,919
Betweenness centrality	France: 0.100 UK: 0.082 USA: 0.066 Australia: 0.063 Canada: 0.053	France: 0.099 UK: 0.083 USA: 0.066 Canada: 0.053 Australia: 0.053	France: 0.095 UK: 0.082 USA: 0.065 Australia: 0.057 Canada: 0.054	France: 0.091 UK: 0.082 USA: 0.064 Australia: 0.061 Canada: 0.052
Betweenness centrality (weighted)	Liechtenstein: 0.335 Ireland: 0.254 Portugal: 0.225 Iceland: 0.200 Bahamas: 0.195	Liechtenstein: 0.264 Chile: 0.205 Costa Rica: 0.194 Iceland: 0.184 Bulgaria: 0.173	Estonia: 0.317 Liechtenstein: 0.226 Costa Rica: 0.202 Iceland: 0.148 Cabo Verde: 0.114	Liechtenstein: 0.268 Guinea: 0.241 Estonia: 0.238 Argentina: 0.226 Chile: 0.148

Note: Syria is ranked 6th by weighted out-degree in 2017 (6.864 million migrants).

4.2 Comparison with directed configuration graph

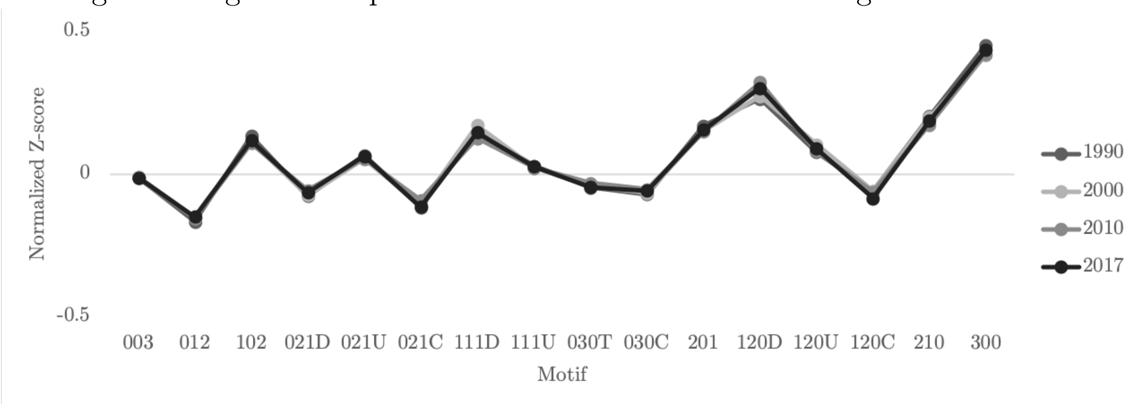
The significance profiles in Figure 3 show the over- and under-representation of each triad on a normalized scale. The following can be observed:

- Motifs with similar characteristics, such as time-variant migrant networks (e.g. 1990, 2000, 2010, 2017) generally move together. This is consistent with what Milo et al. (2004) observed in other disciplines.

- The top motif is type *300* with an average significance profile of 0.43. This is also known as the full-graph motif, where the triangle is fully-connected, supporting the observation earlier on improved global integration. In the 2017 migration network, there were 26,265 occurrences of this motif. The second highest-occurring motif is type *120D*, with an SP of 0.29; and the third highest motif is type *210*, SP of 0.16. Both are similarly structured to a full-graph motif with only one, or two missing edges.

Similarly, in Figure 4, I provide a boxplot of significance profiles generated from the Monte-Carlo test of the configuration model. Note that the scale ranges from -0.01 to +0.01, far removed from the ±0.5 range of the SPs in the immigration network in Figure 3. A full "zoomed-out" boxplot, overlaid with SPs of the immigration network, is provided in appendix B.

Figure 3: Significance profiles for triad census of the immigration network



4.3 Comparison with Erdos-Renyi directed graph

Comparing the migration network's triad census to the Erdos-Renyi random graph did not produce robust results. Interestingly, in a Monte-Carlo test of 49 runs of the random graph, only four triads of the 16 possible types appeared in the triad census. The test results are provided in appendix C.

Figure 4: Monte-carlo test results for triad census of directed configuration graph

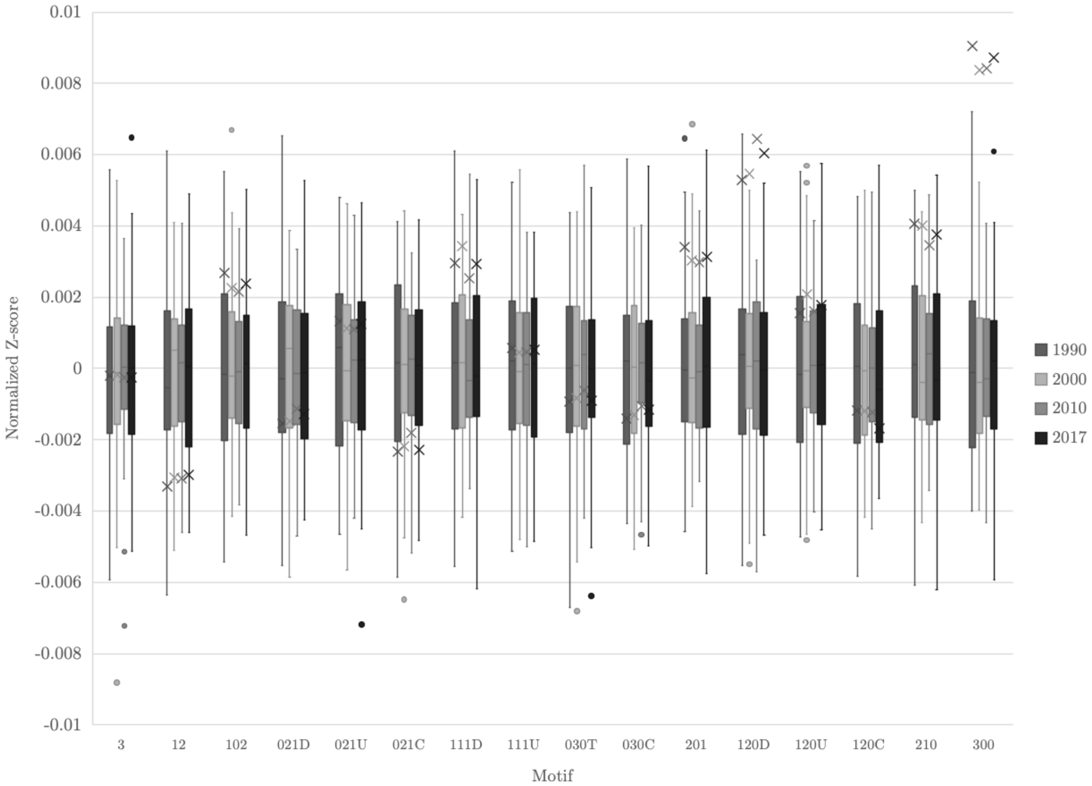


Table 2: Triad census for each of the 16 types, real vs. model network

	Immigration Network				Directed Configuration model (average of 49 runs)			
Motif	**1990**	**2000**	**2010**	**2017**	**1990**	**2000**	**2010**	**2017**
003	934010	917029	874163	871055	944413	929484	891231	887216
012	424112	422299	430246	431082	638694	641266	650093	650609
102	190041	189626	192711	194914	77189	78026	80234	81065
021D	24495	24343	26342	26609	42308	42983	46381	46688
021U	132564	141324	147033	144408	100596	104270	109185	108929
021C	23765	23731	24765	24952	87022	88549	93547	94221
111D	121940	125111	128959	129450	54471	56047	59427	59838
111U	28236	28444	30892	31376	22328	22771	24649	24986
030T	21119	22707	24744	25007	34880	36464	39966	40213
030C	87	88	88	91	4867	5014	5415	5512
201	32584	32330	33666	34198	7314	7482	8058	8234
120D	49908	52704	59729	59299	14148	14795	16288	16403
120U	11911	12403	13520	13586	5010	5250	5771	5902
120C	4932	5080	5280	5372	10622	11058	12081	12291
210	32354	33741	36259	36696	8954	9312	10282	10465
300	22302	23400	25963	26265	1546	1589	1753	1787

4.4 Motif analysis

In this subsection, I provide a closer look at the three prevailing motifs of the immigration network and offer insights on the top decile (10%) of countries that are most associated with each. Recall the structure of each motif, as seen in Figure 5.

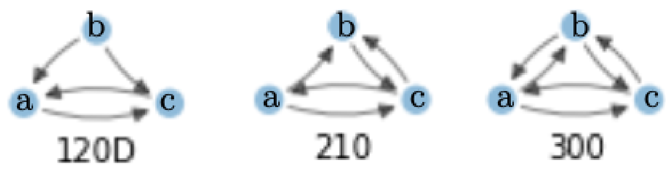


Figure 5: The top three motifs of the immigration network, with node labels for ease of comparison.

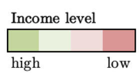
4.4.1 Motif 120D

Using Figure 5 as a guide, we can see that country b is able to migrate to countries a and c, but not vice versa. Meanwhile, countries a and c are able to cross-migrate.

Interestingly, the countries that are highly associated with node b are upper-middle and high-income economies, while those in nodes a and c are a mixed bag. This may suggest that migrants from high-income economies are more mobile and can freely move to low- and middle-income economies but not vice versa. Developing countries, or those that are associated with nodes a and c can freely move across each other, but not necessarily migrate to node b, characterized by countries of high-income. This result holds true both then (1990) and now (2017).

Figure 6: Most represented countries in nodes a, b, c for motif 120D. Numbers in parenthesis indicate the number of motifs for which the country is present.

a	b	c
Austria (2934)	Ireland (1692)	Switzerland (2488)
Australia	Finland	Sweden
Belgium	Italy	Spain
Chile	Greece	UK
Bulgaria	Denmark	Slovenia
Brazil	Netherlands	South Africa
Argentina	Hungary	Norway
Bolivia	Norway	Venezuela
Czechia	Iceland	USA
Canada	France	Netherlands
Denmark	Czechia	Syrian Arab Republic
Cyprus	Chile	Turkey
Finland	Egypt	Portugal
Costa Rica	Costa Rica	Russian Federation
Afghanistan	Spain	Ukraine
Greece	Cyprus	Poland
Algeria	Bulgaria	Viet Nam
China	Russia	Slovakia
France	Sweden	Ireland
Hungary (925)	Iran (739)	Zimbabwe (715)



4.4.2 Motif 210

Motif 210 is almost a complete sub-graph, with one missing arrow from node b to a. Similar to the observation in motif *120D*, it seems that nodes where out-degrees are missing (such as node b in this motif) are associated with developing countries. In Figure 7 we can see that countries in node b comprise of several low-income economies, more than its adjacent nodes a and c.

Figure 7: Most represented countries in nodes a, b, c for motif 210. Numbers in parenthesis indicate the number of motifs for which the country is present.

a	b	c
Australia (1993)	France (1208)	UK (2618)
Canada	Italy	USA
Austria	Venezuela	Turkey
Brazil	Germany	Russia
Belgium	Netherlands	Switzerland
France	Greece	Spain
Argentina	Canada	South Africa
Bulgaria	Ireland	Sweden
Bolivia	Russia	Netherlands
Egypt	Philippines	Philippines
Chile	Japan	Sri Lanka
Germany	Jordan	Portugal
Denmark	Egypt	Slovenia
Costa Rica	Iceland	Thailand
Czechia	Guinea	Romania
Bahamas	Norway	Norway
Belarus	Hungary	Serbia
Finland (609)	Libya (526)	Slovakia (521)

Income level
high low

4.4.3 Motif 300

Also known as the full graph, this motif denotes that each of the three nodes can connect to and from each other. As observed in Figure 8 below, most of the countries in the top decile of this triad type are high-income economies, with the exception of Bolivia and Egypt in the developing category.

Figure 8: Most represented countries in nodes a, b, c for motif 300. Numbers in parenthesis indicate the number of motifs for which the country is present.

a	b	c
Australia (1649)	Italy (947)	UK (1920)
Austria	France	USA
Canada	Germany	Switzerland
Argentina	Netherlands	Spain
Belgium	Greece	Sweden
Brazil	Hungary	Venezuela
Bulgaria	Ireland	Turkey
Chile	Denmark	Russia
France	Norway	South Africa
Bolivia	Finland	Poland
Denmark	Poland	Portugal
Germany	Mexico	Netherlands
Czechia	Russia	Slovenia
Colombia	Canada	Norway
Egypt	Portugal	Peru
Finland	Czechia	Slovakia
Cyprus	Peru	Italy (495)
Croatia (511)	Egypt (558)	

Income level
high low

5 Conclusion

This paper aimed to explore prevailing trends in global migration over time. Particularly, we ought to validate the idea that the flow of immigration is mainly from the developing to the developed world; and from high-risk and conflict-ridden economies to the nations with “safer” borders.

By looking at over-represented structures, or motifs, in the network, one can gain insight on the topology underlying the nature and flow of migration. Of the motifs that emerged in the immigration graph, one can infer that migration to high-income economies is not as flexible as it is between the middle- and low-income economies. One can observe that while middle- and low-income countries enjoy a bi-directional migrant relationship, it seems that high-income economies are more selective of who they take in. High-income economies have mutual dyads to other high-income countries (as motif *300* suggests) but asymmetrical to others (as seen in motifs *120D* and *210*). This result seems to contradict the observation that the flow of migrants mainly comprise those from developing countries and conflict states.

The caveat however tends to be in the weighing of the edges. The triad census only considers unweighted, directed edges. We may be discarding important information, as it can be noted in the network summary statistics that edge weights make a difference in the migration story. Thus, a further improvement to this analysis is in the introduction of weighted motifs, which can enrich our understanding of migrant dynamics.

References

Barder, O., & Krylova, P. (2016, September). *Which Countries Have the Best Migration Policies?* Retrieved 2019-03-30, from https://www.cgdev.org/blog/which-countries-have-best-migration-policies

Hoban, B. (2017, August). *Do immigrants "steal" jobs from American workers?* Retrieved 2019-03-29, from https://www.brookings.edu/blog/brookings-now/2017/08/24/do-immigrants-steal-jobs-from-american-workers/

Koser, K. (2001, November). *When is Migration a Security Issue?* Retrieved 2019-03-29, from https://www.brookings.edu/opinions/when-is-migration-a-security-issue/

Migration Policy Institute. (2017, October). *Largest U.S. Immigrant Groups over Time, 1960-Present.* Retrieved 2019-03-30, from https://www.migrationpolicy.org/programs/data-hub/charts/largest-immigrant-groups-over-time

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., ... Alon, U. (2004, March). Superfamilies of Evolved and Designed Networks. *Science*, *303*(5663), 1538–1542. Retrieved 2019-03-28, from http://science.sciencemag.org/content/303/5663/1538 doi: 10.1126/science.1089167

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002, October). Network Motifs: Simple Building Blocks of Complex Networks. *Science*, *298*(5594), 824–827. Retrieved 2019-03-29, from http://science.sciencemag.org/content/298/5594/824 doi: 10.1126/science.298.5594.824

Squartini, T., van Lelyveld, I., & Garlaschelli, D. (2013, November). Early-warning signals of topological collapse in interbank networks. *Scientific Reports*, *3*, 3357. Retrieved 2019-03-29, from https://www.nature.com/articles/srep03357 doi: 10.1038/srep03357

United Nations database. (2017). *Trends in International Migrant Stock: The 2017 Revision.* United Nations, Department of Economic and Social Affairs, Population Division. Retrieved 2019-03-27, from https://www.un.org/en/development/desa/population/migration/data/estimates2/estimates17.asp

Wasserman, S., & Faust, K. (1994). *Social Network Analysis: Methods and Applications.* Cambridge University Press. (Google-Books-ID: CAm2DpIqRUIC)

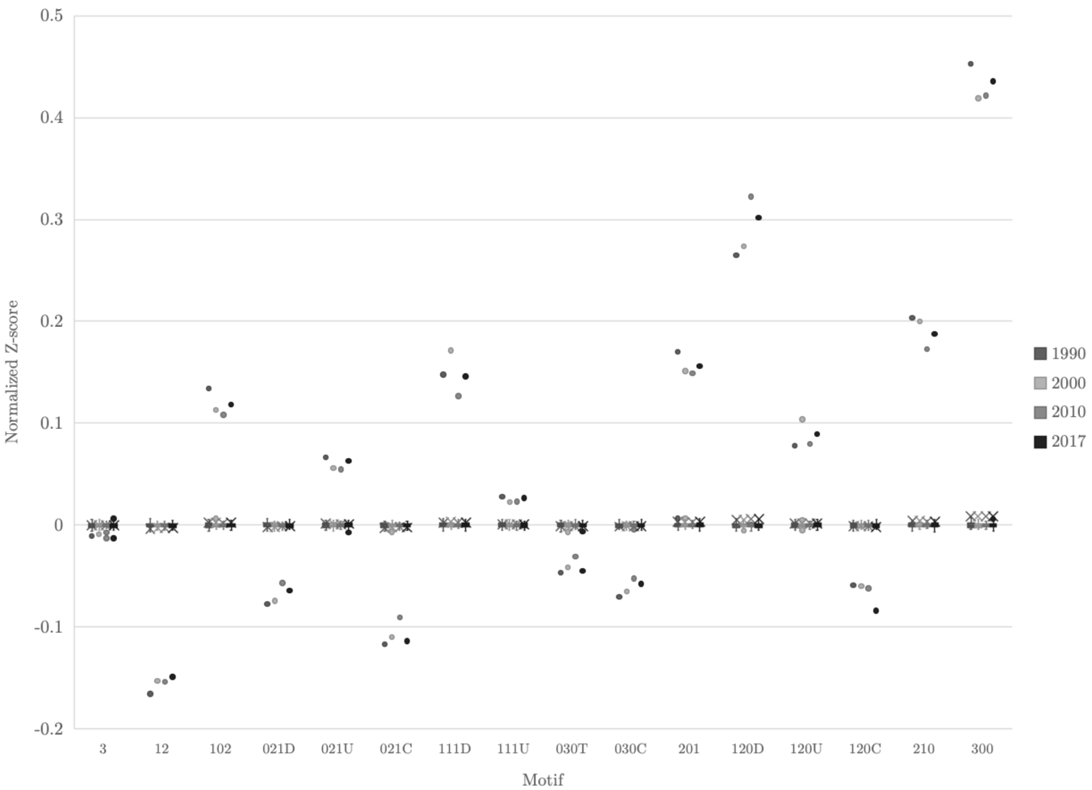
Appendix A Triad census labeling convention

The labeling scheme as described by Wasserman & Faust (1994) is:

- The first number is the number of bidirectional edges, or mutual dyads.
- The second number is the number of single edges, or asymmetric dyads.
- The third number is the number of "non-existent" edges, or null dyads.
- A letter code to distinguish directed variations of the same triad—U for "up," D for "down," C for "circle," and T for "transitive" (i.e., having 2 paths that lead to the same endpoint).

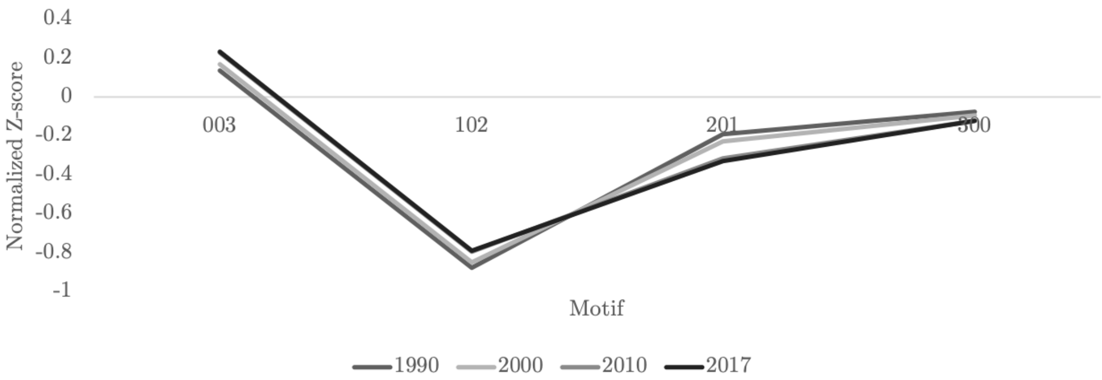
Appendix B Monte-Carlo test: Configuration model, including results from immigration network (N=50)

The boxplot below shows the Monte-Carlo results of the configuration model overlaid with the results from the immigration graph, total N=50. The scale is adjusted from -0.2 to 0.5 to see the full range of the result. It can be observed that the dots, which indicate the results of the immigration graph, are at a notable distance from zero, where the boxplots of the configuration model can be found.



Appendix C Monte-Carlo test: Erdos-Renyi Directed Random Graph

The immigration graph seemed to be a worse fit to an Erdos-Renyi random graph. The 49 iterations of the random graph did not generate triads for 12 of the 16 types to build a complete triad census. The results of the four triads for which I was able to compute significance profiles are provided below.



Appendix D Full country list and UN income classification

Afghanistan	Dem. Rep. Congo	Lesotho	Saint Lucia
Albania	Denmark	Liberia	St Pierre / Miquelon
Algeria	Djibouti	Libya	St Vincent / Grenadines
American Samoa	Dominica	Liechtenstein	Samoa
Andorra	Dominican Republic	Lithuania	San Marino
Angola	Ecuador	Luxembourg	Sao Tome and Principe
Anguilla	Egypt	Madagascar	Saudi Arabia
Antigua and Barbuda	El Salvador	Malawi	Senegal
Argentina	Equatorial Guinea	Malaysia	Serbia
Armenia	Eritrea	Maldives	Seychelles
Aruba	Estonia	Mali	Sierra Leone
Australia	Ethiopia	Malta	Singapore
Austria	Faeroe Islands	Marshall Islands	Sint Maarten
Azerbaijan	Falkland Islands	Martinique	Slovakia
Bahamas	Fiji	Mauritania	Slovenia
Bahrain	Finland	Mauritius	Solomon Islands
Bangladesh	France	Mayotte	Somalia
Barbados	French Guiana	Mexico	South Africa
Belarus	French Polynesia	Micronesia	South Sudan
Belgium	Gabon	Monaco	Spain
Belize	Gambia	Mongolia	Sri Lanka
Benin	Georgia	Montenegro	State of Palestine
Bermuda	Germany	Montserrat	Sudan
Bhutan	Ghana	Morocco	Suriname
Bolivia	Gibraltar	Mozambique	Swaziland
Bosnia and Herzegovina	Greece	Myanmar	Sweden
Botswana	Greenland	Namibia	Switzerland
Brazil	Grenada	Nauru	Syrian Arab Republic
British Virgin Islands	Guadeloupe	Nepal	Tajikistan
Brunei Darussalam	Guam	Netherlands	TFYR Macedonia
Bulgaria	Guatemala	New Caledonia	Thailand
Burkina Faso	Guinea	New Zealand	Timor-Leste
Burundi	Guinea-Bissau	Nicaragua	Togo
Côte d'Ivoire	Guyana	Niger	Tokelau
Cabo Verde	Haiti	Nigeria	Tonga
Cambodia	Holy See	Niue	Trinidad and Tobago
Cameroon	Honduras	Northern Mariana Island	Tunisia
Canada	Hungary	Norway	Turkey
Caribbean Netherlands	Iceland	Oman	Turkmenistan
Cayman Islands	India	Pakistan	Turks and Caicos Islands
Central African Republic	Indonesia	Palau	Tuvalu
Chad	Iran (Islamic Republic of	Panama	Uganda
Channel Islands	Iraq	Papua New Guinea	Ukraine
Chile	Ireland	Paraguay	United Arab Emirates
China	Isle of Man	Peru	United Kingdom
China, Hong Kong SAR	Israel	Philippines	United Rep of Tanzania
China, Macao SAR	Italy	Poland	USA
Colombia	Jamaica	Portugal	US Virgin Islands
Comoros	Japan	Puerto Rico	Uruguay
Congo	Jordan	Qatar	Uzbekistan
Cook Islands	Kazakhstan	Réunion	Vanuatu
Costa Rica	Kenya	Republic of Korea	Venezuela
Croatia	Kiribati	Republic of Moldova	Viet Nam
Cuba	Kuwait	Romania	Wallis / Futuna Islands
Curaçao	Kyrgyzstan	Russian Federation	Western Sahara
Cyprus	Lao People's Democratic	Rwanda	Yemen
Czechia	Latvia	Saint Helena	Zambia
Dem. Rep of Korea	Lebanon	Saint Kitts and Nevis	Zimbabwe

Income level

76	56	52	48
high			low

Appendix E Python code

See the following page for the contents of the Python notebook.

Motifs in Migration Networks (2017 edition)

Step 1. Data Prep

In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_excel("UN_MigrantStockByOriginAndDestination_2017.xlsx", sheet_name='Table 1 clean')
```

In [3]:

```
df = df[~df['Target'].str.contains('developed countries')]
df = df[~df['Target'].str.contains('middle-income countries')]
```

In [4]:

```
df2 = pd.melt(df,id_vars=['Target', 'Year'],var_name='Source', value_name='values')
```

In [5]:

```
df2 = df2.pivot_table(index=['Target', 'Source'], columns='Year', values='values')
df2.head(20)
```

Out[5]:

	Year	1990	1995	2000	2005	2010	2015	2017
Target	**Source**							
Afghanistan	**Pakistan**	8107.0	17225.0	26343.0	37163.0	47984.0	348369.0	95041.0
	Tajikistan	40537.0	36733.0	32929.0	28460.0	23991.0	15031.0	4100.0
	Uzbekistan	2027.0	1836.0	1646.0	1422.0	1199.0	751.0	204.0
Albania	**Canada**	1151.0	1244.0	1337.0	1128.0	920.0	906.0	913.0
	Greece	40087.0	43331.0	46575.0	39315.0	32054.0	31596.0	31871.0
	Italy	11287.0	12200.0	13113.0	11069.0	9025.0	8896.0	8973.0
	TFYR Macedonia	678.0	732.0	787.0	664.0	542.0	534.0	538.0
	Turkey	2489.0	2690.0	2892.0	2441.0	1990.0	1961.0	1978.0
	United States of America	3011.0	3255.0	3498.0	2953.0	2408.0	2373.0	2393.0
Algeria	**France**	2086.0	1321.0	556.0	438.0	481.0	530.0	550.0
	Germany	6544.0	3661.0	778.0	614.0	674.0	743.0	771.0
	Indonesia	1000.0	1223.0	1446.0	1141.0	1253.0	1382.0	1434.0
	Iraq	918.0	4249.0	7579.0	5982.0	6574.0	7256.0	7533.0
	Italy	3143.0	1942.0	740.0	584.0	641.0	707.0	734.0
	Jordan	2286.0	2272.0	2258.0	1782.0	1958.0	2161.0	2243.0
	Kuwait	1420.0	1308.0	1196.0	944.0	1037.0	1144.0	1187.0
	Lebanon	1564.0	1395.0	1225.0	966.0	1061.0	1171.0	1215.0
	Libya	1982.0	2154.0	2325.0	1835.0	2016.0	2225.0	2310.0
	Malaysia	1651.0	1264.0	877.0	692.0	760.0	838.0	870.0
	Russian Federation	6287.0	3545.0	802.0	633.0	695.0	767.0	796.0

In [6]:

```
df2.to_csv("2017_MigrationNetworks.csv", sep=',', encoding='utf-8', index=True)
```

In [ ]:

Part 2. Summary Statistics

In [2]:

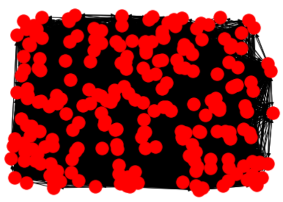
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:

```
g90 = nx.read_graphml("1990.graphml")
g00 = nx.read_graphml("2000.graphml")
g10 = nx.read_graphml("2010.graphml")
g17 = nx.read_graphml("2017.graphml")
```

In [83]:

```
#just want to confirm it's a directed graph
nx.draw_random(g17)
```



In [85]:

```
#Access components of the graph
#g.edges(data=True)
g17['Philippines']['United States of America']['weight']
```

Out[85]:

```
2076253.0
```

Degree

In [259]:

```
years = [g90, g00, g10, g17]
v=['1990','2000','2010','2017']
degree = pd.DataFrame(columns=['year'])
indegree = pd.DataFrame(columns=['year'])
outdegree = pd.DataFrame(columns=['year'])
w_degree = pd.DataFrame(columns=['year'])
w_indegree = pd.DataFrame(columns=['year'])
w_outdegree = pd.DataFrame(columns=['year'])
```

In [29]:

```
for index, g in enumerate(years):
    degree = degree.append(pd.DataFrame(sorted(g.degree(), key=lambda x: x[1])[-5:], columns=['count
ry', 'degree']))
    degree['year'].replace(np.nan,v[index],inplace=True)

    indegree = indegree.append(pd.DataFrame(sorted(g.in_degree(), key=lambda x: x[1])[-10:],
columns=['in_country', 'in_degree']))
    indegree['year'].replace(np.nan,v[index],inplace=True)

    outdegree = outdegree.append(pd.DataFrame(sorted(g.out_degree(), key=lambda x: x[1])[-10:], col
umns=['out_country', 'out_degree']))
    outdegree['year'].replace(np.nan,v[index],inplace=True)

    w_degree = w_degree.append(pd.DataFrame(sorted(g.degree(weight='weight'), key=lambda x: x[1])[-
5:], columns=['country', 'degree']))
    w_degree['year'].replace(np.nan,v[index],inplace=True)

    w_indegree = w_indegree.append(pd.DataFrame(sorted(g.in_degree(weight='weight'), key=lambda x:
x[1])[-10:], columns=['in_country', 'in_degree']))
    w_indegree['year'].replace(np.nan,v[index],inplace=True)

    w_outdegree = w_outdegree.append(pd.DataFrame(sorted(g.out_degree(weight='weight'), key=lambda
x: x[1])[-10:], columns=['out_country', 'out_degree']))
    w_outdegree['year'].replace(np.nan,v[index],inplace=True)

    # average in-degree = average out-degree; the sum is average overall degree
    print(v[index], 'average in/out degree:', np.mean([x[1] for x in sorted(g.in_degree(), key=lamb
da x: x[1])]))
    print(v[index], 'weighted average in/out degree:', np.mean([x[1] for x in sorted(g.in_degree(we
ight='weight'), key=lambda x: x[1])]))
```

```
1990 average in/out degree: 44.4698275862069
1990 weighted average in/out degree: 620928.0431034482
2000 average in/out degree: 45.51293103448276
2000 weighted average in/out degree: 712996.5474137932
2010 average in/out degree: 47.82327586206897
2010 weighted average in/out degree: 912646.6637931034
2017 average in/out degree: 48.00431034482759
2017 weighted average in/out degree: 1065324.0689655172
```

In [240]:

```
pd.concat([indegree, outdegree], axis=1).set_index('year')
```

Out[240]:

	in_country	in_degree	out_country	out_degree
year				
(1990, 1990)	Denmark	196.0	Canada	123.0
(1990, 1990)	United Kingdom	203.0	France	135.0
(1990, 1990)	France	206.0	China	138.0
(1990, 1990)	Greece	209.0	United Kingdom	140.0
(1990, 1990)	Australia	211.0	United States of America	157.0
(2000, 2000)	Ireland	195.0	India	123.0
(2000, 2000)	United Kingdom	205.0	France	136.0
(2000, 2000)	France	206.0	China	139.0
(2000, 2000)	Greece	209.0	United Kingdom	141.0
(2000, 2000)	Chile	210.0	United States of America	158.0
(2010, 2010)	Canada	197.0	India	129.0
(2010, 2010)	Australia	204.0	France	138.0
(2010, 2010)	United Kingdom	205.0	China	143.0
~~**(2010, 2010)**~~	~~France~~	~~206.0~~	~~United Kingdom~~	~~145.0~~
(2010, 2010)	Chile	210.0	United States of America	162.0
(2017, 2017)	Canada	197.0	India	130.0
(2017, 2017)	France	205.0	France	138.0
(2017, 2017)	United Kingdom	205.0	China	143.0
(2017, 2017)	Australia	206.0	United Kingdom	146.0
(2017, 2017)	Chile	210.0	United States of America	162.0

In [245]:

```
pd.concat([w_indegree, w_outdegree], axis=1).set_index('year')
```

Out[245]:

	in_country	in_degree	out_country	out_degree
year				
(1990, 1990)	Iran (Islamic Republic of)	4290497.0	Pakistan	3341574.0
(1990, 1990)	Canada	4327805.0	Italy	3416421.0
(1990, 1990)	Saudi Arabia	4830679.0	United Kingdom	3795662.0
(1990, 1990)	Germany	5601544.0	China	4229860.0
(1990, 1990)	France	5897267.0	Mexico	4394684.0
(1990, 1990)	Pakistan	6203799.0	Bangladesh	5451546.0
(1990, 1990)	Ukraine	6481438.0	Ukraine	5549477.0
(1990, 1990)	India	7362652.0	India	6718862.0
(1990, 1990)	Russian Federation	11516298.0	Afghanistan	6724681.0
(1990, 1990)	United States of America	20134790.0	Russian Federation	12664537.0
(2000, 2000)	Australia	4337890.0	Pakistan	3398405.0
(2000, 2000)	United Kingdom	4692050.0	Kazakhstan	3554534.0
(2000, 2000)	Saudi Arabia	5086745.0	United Kingdom	3866884.0
(2000, 2000)	Ukraine	5137559.0	Afghanistan	4541159.0
(2000, 2000)	Canada	5504699.0	Bangladesh	5435372.0
(2000, 2000)	France	6278718.0	Ukraine	5596463.0
(2000, 2000)	India	6286286.0	China	5786954.0
(2000, 2000)	Germany	8658910.0	India	7978365.0
(2000, 2000)	Russian Federation	11891623.0	Mexico	9562278.0
(2000, 2000)	United States of America	33157941.0	Russian Federation	10734963.0
(2010, 2010)	Australia	5821210.0	United Kingdom	4461711.0
(2010, 2010)	Spain	6278020.0	Philippines	4704919.0
(2010, 2010)	Canada	6751310.0	Afghanistan	4989209.0
(2010, 2010)	United Arab Emirates	7094180.0	Pakistan	5006753.0
(2010, 2010)	France	7196481.0	Ukraine	5458664.0
(2010, 2010)	United Kingdom	7560559.0	Bangladesh	6742845.0
(2010, 2010)	Saudi Arabia	8147064.0	China	8648885.0
(2010, 2010)	Germany	9711410.0	Russian Federation	10213313.0
(2010, 2010)	Russian Federation	11194137.0	Mexico	12413085.0
(2010, 2010)	United States of America	42071829.0	India	13321332.0
(2017, 2017)	Spain	5931689.0	United Kingdom	4921309.0
(2017, 2017)	Australia	7008050.0	Philippines	5680682.0
(2017, 2017)	Canada	7849479.0	Ukraine	5941653.0

year	in_country	in_degree	out_country	out_degree
(2017, 2017)	France	[illegible]	Pakistan	[illegible]
(2017, 2017)	United Arab Emirates	8059782.0	Syrian Arab Republic	6864445.0
(2017, 2017)	United Kingdom	8799334.0	Bangladesh	7499919.0
(2017, 2017)	Russian Federation	11650842.0	China	9962058.0
(2017, 2017)	Saudi Arabia	11774584.0	Russian Federation	10635994.0
(2017, 2017)	Germany	12044115.0	Mexico	12964882.0
(2017, 2017)	United States of America	47412413.0	India	16587720.0

Betweenness centrality | what about for weighted?

In [260]:

```
bc = pd.DataFrame(columns=['year'])
w_bc = pd.DataFrame(columns=['year'])

for index, g in enumerate(years):
    bc = bc.append(pd.DataFrame(sorted(nx.betweenness_centrality(g).items(), key=lambda x: x[1])[-5
:], columns=['country', 'centrality']))
    bc['year'].replace(np.nan,v[index],inplace=True)

    print(v[index],'betweenness centrality:', np.mean([x[1] for x in
sorted(nx.betweenness_centrality(g).items(), key=lambda x: x[1])]))

    w_bc = w_bc.append(pd.DataFrame(sorted(nx.betweenness_centrality(g, weight='weight').items(), k
ey=lambda x: x[1])[-5:], columns=['w_country', 'w_centrality']))
    w_bc['year'].replace(np.nan,v[index],inplace=True)

    print(v[index],'weighted betweenness centrality:', np.mean([x[1] for x in sorted(nx.betweenness
_centrality(g, weight='weight').items(), key=lambda x: x[1])]))

pd.concat([bc, w_bc], axis=1).set_index('year')
```

```
1990 betweenness centrality: 0.003949486295813132
1990 weighted betweenness centrality: 0.016781580542958783
2000 betweenness centrality: 0.003920198991413385
2000 weighted betweenness centrality: 0.015456181154938919
2010 betweenness centrality: 0.003936343516553412
2010 weighted betweenness centrality: 0.01432538214504374
2017 betweenness centrality: 0.003917440630334184
2017 weighted betweenness centrality: 0.015449843878308217
```

Out[260]:

year	centrality	country	w_centrality	w_country
(1990, 1990)	0.052821	Canada	0.195322	Bahamas
(1990, 1990)	0.063491	Australia	0.199747	Iceland
(1990, 1990)	0.066381	United States of America	0.224864	Portugal
(1990, 1990)	0.082139	United Kingdom	0.254294	Ireland
(1990, 1990)	0.100090	France	0.334918	Liechtenstein
(2000, 2000)	0.053031	Australia	0.172861	Bulgaria
(2000, 2000)	0.053177	Canada	0.184429	Iceland
(2000, 2000)	0.066415	United States of America	0.193528	Costa Rica
(2000, 2000)	0.083068	United Kingdom	0.205172	Chile
(2000, 2000)	0.099177	France	0.264325	Liechtenstein
(2010, 2010)	0.053564	Canada	0.114614	Cabo Verde
(2010, 2010)	0.056822	Australia	0.147746	Iceland
(2010, 2010)	0.065300	United States of America	0.202146	Costa Rica
(2010, 2010)	0.082205	United Kingdom	0.225701	Liechtenstein
(2010, 2010)	0.094714	France	0.317294	Estonia
(2017, 2017)	0.052164	Canada	0.147762	Chile
(2017, 2017)	0.060196	Australia	0.225801	Argentina
(2017, 2017)	0.064038	United States of America	0.238287	Estonia
(2017, 2017)	0.082109	United Kingdom	0.240754	Guinea
(2017, 2017)	0.090593	France	0.267688	Liechtenstein

In [262]:

```
#nx.betweenness_centrality(g)
```

Clustering coefficient

The global cc (transitivity) gives an overall indication of the clustering in the network, whereas the local gives an indication of the embeddedness of single nodes.

In [262]:

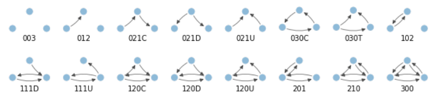
```
for index, g in enumerate(years):
    #global
    print(v[index], 'transitivity:', nx.transitivity(g))

    #local = proportion of friends that are friends themselves
    print(v[index], 'local clustering average', nx.average_clustering(g))
```

```
1990 transitivity: 0.6008067317103976
1990 local clustering average 0.630758902363683
2000 transitivity: 0.6090046815566176
2000 local clustering average 0.6330730135082913
2010 transitivity: 0.615857302429489
2010 local clustering average 0.6323039318145905
2017 transitivity: 0.6143444438292454
2017 local clustering average 0.6323568965240542
```

Part 3. Motif detection

In [263]:

```
from triadic_census import triadic, draw_triads
```



In [54]:

```
tc = pd.DataFrame(index=['003', '012', '102', '021D', '021U', '021C', '111D', '111U', '030T',
       '030C', '201', '120D', '120U', '120C', '210', '300'])

for index, g in enumerate(years):
    temp = pd.DataFrame.from_dict(nx.triadic_census(g), orient='index')
    temp.rename(columns={0:v[index]}, inplace=True)
    tc = tc.join(temp)
```

In [172]:

```
tc
```

Out[172]:

	1990	2000	2010	2017
003	934010	917029	874163	871055
012	424112	422299	430246	431082
102	190041	189626	192711	194914
021D	24495	24343	26342	26609
021U	132564	141324	147033	144408
021C	23765	23731	24765	24952
111D	121940	125111	128959	129450
111U	28236	28444	30892	31376
030T	21119	22707	24744	25007
030C	87	88	88	91
201	32584	32330	33666	34198
120D	49908	52704	59729	59299
120U	11911	12403	13520	13586
120C	4932	5080	5280	5372
210	32354	33741	36259	36696
300	22302	23400	25963	26265

Part 4. Model Fitting

To compare whether these results are abnormal = Monte Carlo test of a configuration model or an Erdos-Renyi

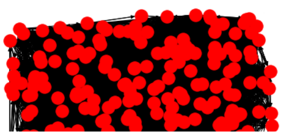
Configuration Model

In [173]:

```
montecarlo_cm = tc

#Degree distributions
#Unweighted
for index, g in enumerate(years):
    din=[item[1] for item in g.in_degree()]
    dout=[item[1] for item in g.out_degree()]

    for run in range(49):
        Cg=nx.directed_configuration_model(din,dout)
        temp = pd.DataFrame.from_dict(nx.triadic_census(Cg), orient='index')
        temp.rename(columns={0:v[index]+'cm'+str(run)}, inplace=True)
        montecarlo_cm = montecarlo_cm.join(temp)

#Weighted
#     din_w=[item[1] for item in g.in_degree(weight='weight')]
#     dout_w=[item[1] for item in g.out_degree(weight='weight')]
#     Cgw=nx.directed_configuration_model(din,dout)
```

In [60]:

```
nx.draw_random(Cg)
```



In [59]:

```
montecarlo_cm
```

Out[59]:

	1990	2000	2010	2017	1990cm0	1990cm1	2000cm0	2000cm1	2010cm0	2010cm1	2017cm0	2017cm1
003	934010	917029	874163	871055	944057	944061	925037	931768	886444	891748	890211	889885
012	424112	422299	430246	431082	642727	639727	646898	640377	650511	650576	646644	645240
102	190041	189626	192711	194914	76565	77959	74551	79760	78027	80688	81091	83508
021D	24495	24343	26342	26609	42319	42163	44011	42884	47230	45882	45860	45847
021U	132564	141324	147033	144408	100887	99600	104776	103257	111132	108502	108835	107703
021C	23765	23731	24765	24952	85713	86387	90722	87405	94044	92631	95682	93128
111D	121940	125111	128959	129450	53754	55040	54868	56151	59750	60009	59112	61879
111U	28236	28444	30892	31376	21556	22150	22032	23175	24557	24802	25631	25416
030T	21119	22707	24744	25007	35012	35039	37472	35730	41225	40069	40981	39161
030C	87	88	88	91	4818	4789	5215	4876	5736	5237	5733	5554
201	32584	32330	33666	34198	7076	7555	7392	7390	8419	8272	7907	8926
120D	49908	52704	59729	59299	13831	13767	14822	14610	16543	16209	16316	16619
120U	11911	12403	13520	13586	5185	4978	5061	5353	5886	5846	6004	5979
120C	4932	5080	5280	5372	10524	10607	10825	10873	12477	12000	12395	12584
210	32354	33741	36259	36696	8765	9045	9097	9255	10635	10265	10277	11010
300	22302	23400	25963	26265	1571	1493	1581	1496	1744	1624	1681	1921

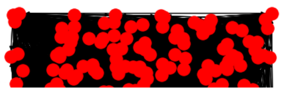
Erdos-Renyi

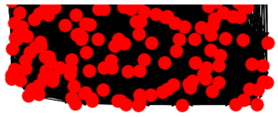
In [174]:

```
montecarlo = montecarlo_cm

for index, g in enumerate(years):
    n = g.number_of_nodes()
    p = 2*g.number_of_edges()/float(n*(n-1))
    p #edge probability not probability

    for run in range(49):
        R = nx.fast_gnp_random_graph(n,p)
        Rd = R.to_directed()
        temp = pd.DataFrame.from_dict(nx.triadic_census(Rd), orient='index')
        temp.rename(columns={0:v[index]+'er'+str(run)}, inplace=True)
        montecarlo = montecarlo.join(temp)
```

In [62]:

```
nx.draw_random(Rd)
```





In [278]:

```python
#Prepare for analysis
sp = montecarlo.T
sp['decade']=sp.index

mc_list = ['1990cm', '2000cm', '2010cm', '2017cm','1990er', '2000er', '2010er', '2017er']

for decade in mc_list:
    for i in range(0,49):
        if sp.index.contains(decade+str(i)):
            sp['decade'][np.where(sp.index==decade+str(i))[0]] = decade
```

In [243]:

```python
sp
```

Out[243]:

	003	012	102	021D	021U	021C	111D	111U	030T	030C	201	120D	120U	120C	210	300
1990	934010	424112	190041	24495	132564	23765	121940	28236	21119	87	32584	49908	11911	4932	32354	22302
2000	917029	422299	189626	24343	141324	23731	125111	28444	22707	88	32330	52704	12403	5080	33741	23400
2010	874163	430246	192711	26342	147033	24765	128959	30892	24744	88	33666	59729	13520	5280	36259	25963
2017	871055	431082	194914	26609	144408	24952	129450	31376	25007	91	34198	59299	13586	5372	36696	26265
1990cm0	945006	635617	79025	41535	100881	85767	55946	22855	33922	4571	7554	15037	4821	10614	9333	1876
1990cm1	947217	636498	77228	41610	100118	86965	54543	22637	34954	4911	7544	13691	5199	10676	9049	1520
1990cm2	945856	636934	80203	42894	98371	84629	55950	22193	34261	4638	8021	14316	4718	10401	9278	1697
1990cm3	949678	636742	79458	42147	99916	84819	54541	22407	33690	4667	7146	13729	5060	10220	8637	1503
1990cm4	946617	633882	79844	41894	100535	85172	56268	22937	34089	4723	7721	14196	5001	10699	9263	1519
1990cm5	948424	636973	80203	41973	98660	85360	55086	23079	32894	4613	7708	13782	4723	10286	8963	1633
1990cm6	942927	640767	77761	42203	99441	87426	53259	22418	35071	4867	7336	14203	5188	10762	9196	1535
1990cm7	942630	634255	80529	42105	100973	86089	56117	23420	34604	4902	7848	14003	5127	10720	9416	1622
1990cm8	948878	632739	80726	41030	99107	85746	55621	23184	34594	4720	7948	13418	5501	10786	8978	1384
1990cm9	944958	643527	74067	42875	101142	88553	52036	21621	34898	4846	6691	13944	4864	10301	8496	1541
1990cm10	943336	642490	75945	42471	100007	87956	53219	21729	35114	5135	7270	14025	4891	10641	8723	1408
1990cm11	943364	640281	76576	42261	101516	87291	53772	22026	35135	4843	7018	14345	4780	10518	8995	1639
1990cm12	942882	643162	77834	42785	98576	86497	54378	21983	34562	4690	7203	14126	4910	10387	8839	1546
1990cm13	945179	640739	75485	42245	101579	87831	53175	21463	35928	4893	6632	13883	4848	10558	8542	1380
1990cm14	940391	639740	77591	42001	101804	87300	54929	21972	35203	4895	7409	14263	5036	10865	9257	1704
1990cm15	944486	637156	76860	42289	100975	86922	55447	22373	34977	4839	7274	14245	5263	10598	9122	1534
1990cm16	946236	640772	76420	42754	99558	87040	53606	21965	34545	4860	7265	13996	4849	10430	8638	1426
1990cm17	949962	637684	76770	41645	99460	86880	53686	22097	34379	4790	7318	13903	5191	10402	8764	1429
1990cm18	947018	635632	80307	41580	99222	85306	55322	22940	34016	4587	7748	13859	5307	10746	9277	1493
1990cm19	942599	644156	74529	43341	101054	88457	52670	21702	34985	4897	6909	13983	4622	10370	8512	1574
1990cm20	943841	643258	73660	43376	101357	89253	51929	21986	35302	5149	6730	13401	4943	10424	8376	1375
1990cm21	942402	639957	74928	42529	101376	88817	54139	22289	35794	5267	7194	13795	4992	10798	8688	1395
1990cm22	944601	636320	77511	42231	100884	87167	54812	22793	35208	4934	7188	14403	5118	10870	8893	1427
1990cm23	940900	646599	72618	43815	102098	88923	52317	21257	35801	5021	6766	13917	4591	10060	8253	1424
1990cm24	941866	639956	76498	42428	100632	87331	54853	22060	35632	4963	7280	14228	5148	10896	8980	1609
1990cm25	943545	643130	75202	43163	99304	88256	53699	21885	34129	4902	7304	14198	4660	10501	8849	1633
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2017er19	415930	0	876767	0	0	0	0	0	0	0	617266	0	0	0	0	144397
2017er20	406510	0	875221	0	0	0	0	0	0	0	624008	0	0	0	0	148621
2017er21	400926	0	871749	0	0	0	0	0	0	0	629994	0	0	0	0	151691
2017er22	409259	0	874846	0	0	0	0	0	0	0	622491	0	0	0	0	147764
2017er23	416435	0	877301	0	0	0	0	0	0	0	616293	0	0	0	0	144331
2017er24	411450	0	875353	0	0	0	0	0	0	0	621574	0	0	0	0	145983
2017er25	400801	0	871458	0	0	0	0	0	0	0	630491	0	0	0	0	151610
2017er26	410242	0	874537	0	0	0	0	0	0	0	622230	0	0	0	0	147351
2017er27	405095	0	872870	0	0	0	0	0	0	0	626745	0	0	0	0	149650
2017er28	400359	0	869666	0	0	0	0	0	0	0	631031	0	0	0	0	153304
2017er29	406202	0	873153	0	0	0	0	0	0	0	626078	0	0	0	0	148927
2017er30	413247	0	874932	0	0	0	0	0	0	0	620015	0	0	0	0	146166
2017er31	412707	0	875599	0	0	0	0	0	0	0	619841	0	0	0	0	146213
2017er32	412209	0	876710	0	0	0	0	0	0	0	619573	0	0	0	0	145868
2017er33	407926	0	873346	0	0	0	0	0	0	0	624200	0	0	0	0	148888
2017er34	413401	0	877071	0	0	0	0	0	0	0	618495	0	0	0	0	145393
2017er35	416833	0	878342	0	0	0	0	0	0	0	615317	0	0	0	0	143868
2017er36	410004	0	874433	0	0	0	0	0	0	0	622232	0	0	0	0	147691
2017er37	400908	0	869916	0	0	0	0	0	0	0	631184	0	0	0	0	152352
2017er38	406734	0	872343	0	0	0	0	0	0	0	625642	0	0	0	0	149641
2017er39	417115	0	877149	0	0	0	0	0	0	0	616397	0	0	0	0	143699
2017er40	411627	0	875445	0	0	0	0	0	0	0	620169	0	0	0	0	147119
2017er41	421512	0	879821	0	0	0	0	0	0	0	610972	0	0	0	0	142055
2017er42	400632	0	871238	0	0	0	0	0	0	0	630748	0	0	0	0	151742
2017er43	407805	0	874917	0	0	0	0	0	0	0	624181	0	0	0	0	147457
2017er44	413326	0	877065	0	0	0	0	0	0	0	618502	0	0	0	0	145467
2017er45	401731	0	869967	0	0	0	0	0	0	0	630453	0	0	0	0	152209
2017er46	414576	0	878024	0	0	0	0	0	0	0	616974	0	0	0	0	144786
2017er47	394957	0	869219	0	0	0	0	0	0	0	635481	0	0	0	0	154703
2017er48	400500	0	871234	0	0	0	0	0	0	0	630462	0	0	0	0	152164

396 rows × 17 columns

In [176]:

```python
#Group by means and SD
sp.groupby(['decade']).mean()
```

Out[176]:

decade	003	012	102	021D	021U	021C	111D	111U	
1990	934010.000000	424112.000000	190041.000000	24495.000000	132564.000000	23765.000000	121940.000000	28236.000000	2111
1990cm	944412.673469	638693.571429	77189.204082	42307.755102	100595.551020	87022.469388	54470.612245	22327.979592	3488
1990er	477190.285714	0.000000	897188.244898	0.000000	0.000000	0.000000	0.000000	0.000000	0
2000	917029.000000	422299.000000	189626.000000	24343.000000	141324.000000	23731.000000	125111.000000	28444.000000	2270
2000cm	929483.877551	641265.612245	78025.775510	42983.000000	104270.408163	88548.775510	56047.408163	22770.979592	3646
2000er	457647.408163	0.000000	891797.367347	0.000000	0.000000	0.000000	0.000000	0.000000	0
2010	874163.000000	430246.000000	192711.000000	26342.000000	147033.000000	24765.000000	128959.000000	30892.000000	2474
2010cm	891230.734694	650093.102041	80234.367347	46381.428571	109184.632653	93547.163265	59426.693878	24648.734694	3996
2010er	413965.693878	0.000000	876284.489796	0.000000	0.000000	0.000000	0.000000	0.000000	0
2017	871055.000000	431082.000000	194914.000000	26609.000000	144408.000000	24952.000000	129450.000000	31376.000000	2500
2017cm	887215.653061	650608.530612	81065.265306	46688.040816	108929.102041	94220.673469	59838.469388	24985.857143	4021
2017er	407380.775510	0.000000	873752.122449	0.000000	0.000000	0.000000	0.000000	0.000000	0

In [177]:

```python
sp.groupby(['decade']).std()
```

Out[177]:

decade	003	012	102	021D	021U	021C	111D	111U	030T	030
1990	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1990cm	2479.825577	3231.201737	2095.416028	575.301208	1195.654512	1352.338254	1139.139145	521.636946	739.006390	169.64324
1990er	5649.769907	0.000000	1364.639338	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
2000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2000cm	3231.150528	3272.178198	2254.829486	574.645724	1512.756671	1351.828457	920.310159	572.553145	763.334012	173.20763
2000er	6891.867390	0.000000	2080.739879	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
2010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2010cm	2676.907107	2889.333792	2099.524983	712.710115	1399.293463	1539.613714	1107.651536	545.600043	998.824663	205.66487
2010er	6435.244945	0.000000	2754.018731	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
2017	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2017cm	3015.735217	3497.044312	2275.091177	744.488610	1330.264483	1444.301033	1128.359098	567.196284	800.338068	223.27977
2017er	6372.798804	0.000000	2792.151557	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000

Z-score

In [178]:

```python
sp.groupby(['decade']).mean().to_csv("montecarlo-means.csv", sep=',', encoding='utf-8', index=True
)
sp.groupby(['decade']).std().to_csv("montecarlo-stdevs.csv", sep=',', encoding='utf-8', index=True)
```

In [217]:

```python
results = pd.read_excel("montecarlo-charts.xlsx", sheet_name="Calculations", index_col="Motif")
results
```

Out[217]:

Motif	1990	1990cm	1990cm-std	z score	SP	1990er	1990er-std	z score.1	SP.1	2000	...	
003	934010.0	944412.673469	2479.825577	-4.194921	-0.010454	477190.285714	5649.769907	80.856340	0.136266	917029.0	...	0.2
012	424112.0	638693.571429	3231.201737	-66.409215	-0.165499	0.000000	0.000000	NaN	0.000000	422299.0	...	0.0
102	190041.0	77189.204082	2095.416028	53.856511	0.134217	897188.244898	1364.639338	-518.193508	-0.873305	189626.0	...	0.
021D	24495.0	42307.755102	575.301208	-30.962485	-0.077162	0.000000	0.000000	NaN	0.000000	24343.0	...	0.0
021U	132564.0	100595.551020	1195.654512	26.737196	0.066632	0.000000	0.000000	NaN	0.000000	141324.0	...	0.0
021C	23765.0	87022.469388	1352.338254	-46.776366	-0.116572	0.000000	0.000000	NaN	0.000000	23731.0	...	0.0
111D	121940.0	54470.612245	1139.139145	59.228399	0.147604	0.000000	0.000000	NaN	0.000000	125111.0	...	0.0
111U	28236.0	22327.979592	521.636946	11.325924	0.028225	0.000000	0.000000	NaN	0.000000	28444.0	...	0.0
030T	21119.0	34880.183673	739.006390	-18.621197	-0.046406	0.000000	0.000000	NaN	0.000000	22707.0	...	0.0
030C	87.0	4866.591837	169.643243	-28.174372	-0.070214	0.000000	0.000000	NaN	0.000000	88.0	...	0.0
201	32584.0	7313.897959	370.440799	68.216304	0.170003	562481.224490	4711.387239	-112.471592	-0.189547	32330.0	...	0.
120D	49908.0	14147.734694	336.514102	106.266766	0.264829	0.000000	0.000000	NaN	0.000000	52704.0	...	0.0
120U	11911.0	5010.265306	220.932378	31.234601	0.077840	0.000000	0.000000	NaN	0.000000	12403.0	...	0.0
120C	4932.0	10621.551020	240.395426	-23.667468	-0.058982	0.000000	0.000000	NaN	0.000000	5080.0	...	0.0
210	32354.0	8954.204082	286.863061	81.571311	0.203285	0.000000	0.000000	NaN	0.000000	33741.0	...	0.0
300	22302.0	1545.755102	114.274620	181.634775	0.452655	117500.244898	2185.353276	-43.561948	-0.073414	23400.0	...	0.
∑ Z	NaN	NaN	NaN	401.265762	NaN	NaN	NaN	593.370708	NaN	NaN	...	

17 rows × 36 columns

In [257]:

```python
import matplotlib.ticker as ticker

plt.figure(0, figsize=(9, 3))
plt.plot(results.index, results['SP'])
plt.plot(results.index, results['SP.2'])
plt.plot(results.index, results['SP.4'])
plt.plot(results.index, results['SP.6'])
plt.ylim(-0.5,0.5)
#plt.yscale('log')
```

Out[257]:

(-0.5, 0.5)

Part 5. Checking interesting motifs: 210, 120D, 300

In [222]:

```python
G = g17
#: The integer codes representing each type of triad.
#:
#: Triads that are the same up to symmetry have the same code.
TRICODES = (1, 2, 2, 3, 2, 4, 6, 8, 2, 6, 5, 7, 3, 8, 7, 11, 2, 6, 4, 8, 5, 9,
            9, 13, 6, 10, 9, 14, 7, 14, 12, 15, 2, 5, 6, 7, 6, 9, 10, 14, 4, 9,
            9, 12, 8, 13, 14, 15, 3, 7, 8, 11, 7, 12, 14, 15, 8, 14, 13, 15,
            11, 15, 15, 16)
#: The names of each type of triad. The order of the elements is
#: important: it corresponds to the tricodes given in :data:`TRICODES`.
TRIAD_NAMES = ('003', '012', '102', '021D', '021U', '021C', '111D', '111U',
               '030T', '030C', '201', '120D', '120U', '120C', '210', '300')

#: A dictionary mapping triad code to triad name.
TRICODE_TO_NAME = {i: TRIAD_NAMES[code - 1] for i, code in enumerate(TRICODES)}
```

```python
def _tricode(G, v, u, w):
    """Returns the integer code of the given triad.

    This is some fancy magic that comes from Batagelj and Mrvar's paper. It
    treats each edge joining a pair of `v`, `u`, and `w` as a bit in
    the binary representation of an integer.

    """
    combos = ((v, u, 1), (u, v, 2), (v, w, 4), (w, v, 8), (u, w, 16),
              (w, u, 32))
    return sum(x for u, v, x in combos if v in G[u])


census = {name: set([]) for name in TRIAD_NAMES}
n = len(G)
m = {v: i for i, v in enumerate(G)}
for v in G:
    vnbrs = set(G.pred[v]) | set(G.succ[v])
    for u in vnbrs:
        if m[u] <= m[v]:
            continue
        neighbors = (vnbrs | set(G.succ[u]) | set(G.pred[u])) - {u, v}
        # Calculate dyadic triads instead of counting them.
        for w in neighbors:
            if v in G[u] and u in G[v]:
                census['102'].add(tuple(sorted([u, v, w])))
            else:
                census['012'].add(tuple(sorted([u, v, w])))
        # Count connected triads.
        for w in neighbors:
            if m[u] < m[w] or (m[v] < m[w] < m[u] and
                               v not in G.pred[w] and
                               v not in G.succ[w]):
                code = _tricode(G, v, u, w)
                census[TRICODE_TO_NAME[code]].add(tuple(sorted([u, v, w])))

# null triads, I implemented them manually because the original algorithm computes
# them as _number_of_all_possible_triads_ - _number_of_all_found_triads_
for v in G:
    vnbrs = set(G.pred[v]) | set(G.succ[v])
    not_vnbrs = set(G.nodes()) - vnbrs
    for u in not_vnbrs:
        unbrs = set(G.pred[u]) | set(G.succ[u])
        not_unbrs = set(G.nodes()) - unbrs
        for w in not_unbrs:
            wnbrs = set(G.pred[w]) | set(G.succ[w])
            if v not in wnbrs and len(set([u, v, w])) == 3:
                census['003'].add(tuple(sorted([u, v, w])))
```

In [263]:

```python
#analyzing which countries appear the most

import collections
triads = ['201']#, '300', '120D']

for i in triads:
    a = [item[0] for item in census[i]]
    b = [item[1] for item in census[i]]
    c = [item[2] for item in census[i]]

    a = pd.DataFrame.from_dict(collections.Counter(a), orient ='index')
    b = pd.DataFrame.from_dict(collections.Counter(b), orient ='index')
    c = pd.DataFrame.from_dict(collections.Counter(c), orient ='index')
    #pd.concat([a, b, c], axis=1, sort=True).to_excel("census_counts.xlsx", sheet_name=i)
```

In [237]:

```python
print(len(census['300']))
```

```
26265
```

In [230]:

```python
nx.triadic_census(g17)
```

Out[230]:

```
{'003': 871055,
 '012': 431082,
 '102': 194914,
 '021D': 26609,
 '021U': 144408,
 '021C': 24952,
 '111D': 129450,
 '111U': 31376,
 '030T': 25007,
 '030C': 91,
 '201': 34198,
 '120D': 59299,
 '120U': 13586,
 '120C': 5372,
 '210': 36696,
 '300': 26265}
```

In [274]:

```python
#c.sort_values(by=[0], ascending=False)
```

In [275]:

```python
census['201'] #forbidden triad
```

Out[275]:

```
{('France', 'Nicaragua', 'Qatar'),
 ('Australia', 'Bulgaria', 'Republic of Korea'),
 ('Lebanon', 'Malaysia', 'Philippines'),
 ('Andorra', 'Argentina', 'Hungary'),
 ('Burundi', 'France', 'Turkey'),
 ('Bhutan', 'Italy', 'Seychelles'),
 ('Germany', 'Tunisia', 'Turkmenistan'),
 ('Honduras', 'Malta', 'Spain'),
 ('Brazil', 'Guinea', 'Suriname'),
 ('Indonesia', 'Philippines', 'Puerto Rico'),
 ('Cyprus', 'Saint Helena', 'South Africa'),
 ('Cabo Verde', 'France', 'Thailand'),
 ('Armenia', 'India', 'Turkey'),
 ('Paraguay', 'Samoa', 'United Kingdom'),
 ('Maldives', 'Mongolia', 'United Kingdom'),
 ('Croatia', 'Egypt', 'Qatar'),
 ('Guyana', 'United States of America', 'Uruguay'),
 ('Germany', 'Guinea', 'Turkmenistan'),
 ('Central African Republic', 'Egypt', 'Namibia'),
 ('France', 'Haiti', 'Mongolia'),
 ('France', 'Lithuania', 'Republic of Korea'),
 ('Azerbaijan', 'Cuba', 'Russian Federation'),
 ('Germany', 'Paraguay', 'Trinidad and Tobago'),
 ('Romania', 'Seychelles', 'Spain'),
 ('British Virgin Islands', 'Canada', 'New Zealand'),
 ('Cabo Verde', 'Egypt', 'Spain'),
 ('Burundi', 'El Salvador', 'France'),
 ('Liberia', 'Namibia', 'United Kingdom'),
 ('Andorra', 'Chile', 'Croatia'),
 ('Bolivia (Plurinational State of)', 'Liechtenstein', 'United Kingdom'),
 ('Colombia', 'France', 'San Marino'),
 ('Australia', 'Isle of Man', 'Maldives'),
 ('Canada', 'Norway', 'Papua New Guinea'),
 ('Bermuda', 'Brunei Darussalam', 'United Kingdom'),
 ('Algeria', 'Italy', 'Mauritius'),
 ('Canada', 'Mauritius', 'Papua New Guinea'),
 ('France', 'India', 'Morocco'),
 ('Australia', 'Fiji', 'New Caledonia'),
 ('Canada', 'Gabon', 'Mauritius'),
 ('Denmark', 'Faeroe Islands', 'Slovenia'),
 ('Lithuania', 'New Zealand', 'United States of America'),
 ('Canada', 'Nicaragua', 'Sierra Leone'),
 ('Antigua and Barbuda', 'Georgia', 'United Kingdom'),
```

```
 ('Bahrain', 'Nicaragua', 'United Kingdom'),
 ('Isle of Man', 'Liechtenstein', 'United Kingdom'),
 ('Canada', 'Gabon', 'Zambia'),
 ('Sierra Leone', 'Thailand', 'United Kingdom'),
 ('Austria', 'Belarus', 'Colombia'),
 ('Australia', 'Iceland', 'Samoa'),
 ('Argentina', 'Liechtenstein', 'Namibia'),
 ('Caribbean Netherlands', 'France', 'Thailand'),
 ('Guinea', 'Slovakia', 'South Africa'),
 ('Puerto Rico', 'United States of America', 'Zambia'),
 ('France', 'Haiti', 'Tunisia'),
 ('El Salvador', 'France', 'Ghana'),
 ('Montenegro', 'Slovakia', 'South Africa'),
 ('Guinea-Bissau', 'Latvia', 'Portugal'),
 ('Dominican Republic', 'Guatemala', 'Martinique'),
 ('Colombia', 'Tajikistan', 'United States of America'),
 ('France', 'Senegal', 'Seychelles'),
 ('Philippines', 'Spain', 'Timor-Leste'),
 ('Belarus', 'Mexico', 'Spain'),
 ('Brunei Darussalam', 'Namibia', 'United Kingdom'),
 ('Central African Republic', 'Seychelles', 'South Africa'),
 ('Botswana', 'Liberia', 'Sweden'),
 ('Ireland', 'Mongolia', 'Papua New Guinea'),
 ('Guatemala', 'Romania', 'Switzerland'),
 ('Bahrain', 'Maldives', 'Sri Lanka'),
 ('Maldives', 'United Kingdom', 'Vanuatu'),
 ('Caribbean Netherlands', 'Costa Rica', 'France'),
 ('Australia', "Dem. People's Republic of Korea", 'Swaziland'),
 ('Puerto Rico', 'Uruguay', 'Venezuela (Bolivarian Republic of)'),
 ('Chile', 'New Zealand', 'Paraguay'),
 ('Malta', 'Mozambique', 'Portugal'),
 ('Central African Republic', 'Luxembourg', 'Sweden'),
 ('Ireland', 'Mongolia', 'Venezuela (Bolivarian Republic of)'),
 ('Antigua and Barbuda', 'United Kingdom', 'Zambia'),
 ('Bulgaria', 'Costa Rica', 'Namibia'),
 ('Honduras', 'United Kingdom', 'Viet Nam'),
 ('Cayman Islands', 'French Polynesia', 'United Kingdom'),
 ('Argentina', 'Canada', 'Mauritius'),
 ('Cabo Verde', 'Paraguay', 'United States of America'),
 ('Germany', 'Kazakhstan', 'Nicaragua'),
 ('Bahamas', 'Croatia', 'Netherlands'),
 ('Canada', 'El Salvador', 'Samoa'),
 ('Albania', 'Algeria', 'United States of America'),
 ('Australia', 'Cayman Islands', 'Ecuador'),
 ('Lithuania', 'Thailand', 'United States of America'),
 ('Georgia', 'Malawi', 'United Kingdom'),
 ('Georgia', 'Jordan', 'Romania'),
 ('Croatia', 'Eritrea', 'United Kingdom'),
 ('Belize', 'Egypt', 'Mexico'),
 ('Argentina', 'Austria', 'Tajikistan'),
 ('Canada', 'China, Hong Kong SAR', 'Guyana'),
 ('Brazil', 'Egypt', 'Honduras'),
 ('Croatia', 'Mexico', 'Puerto Rico'),
 ('Argentina', 'Canada', 'French Polynesia'),
 ('China, Hong Kong SAR', 'France', 'Western Sahara'),
 ('Canada', 'Grenada', 'Guinea-Bissau'),
 ('Republic of Korea', 'Russian Federation', 'San Marino'),
 ('Poland', 'Sierra Leone', 'United Kingdom'),
 ('Cameroon', 'United Arab Emirates', 'United States of America'),
 ('Austria', 'Bahamas', 'Georgia'),
 ('Iceland', 'Italy', 'Montenegro'),
 ('Eritrea', 'France', 'Panama'),
 ('Bosnia and Herzegovina', 'Ecuador', 'Germany'),
 ('Bulgaria', 'Netherlands', 'Saint Lucia'),
 ('Indonesia', 'Lithuania', 'United States of America'),
 ('Germany', 'Panama', 'Tajikistan'),
 ('Bahamas', 'France', 'Luxembourg'),
 ('Antigua and Barbuda', 'Sri Lanka', 'United Kingdom'),
 ('Cameroon', 'France', 'San Marino'),
 ('Canada', 'Ireland', 'Saint Pierre and Miquelon'),
 ('Algeria', 'United Kingdom', 'Uruguay'),
 ('Jordan', 'Mali', 'Russian Federation'),
 ('Brazil', 'Falkland Islands (Malvinas)', 'United Kingdom'),
 ('Cayman Islands', 'Ireland', 'Mongolia'),
 ('Belgium', 'Libya', 'Nicaragua'),
 ('Germany', 'Kuwait', 'Sri Lanka'),
 ('Belgium', 'Hungary', 'Seychelles'),
```

```
 ('Czechia', 'New Zealand', 'Slovenia'),
 ('Guinea', 'Russian Federation', 'Slovakia'),
 ('Bulgaria', 'Central African Republic', 'Cyprus'),
 ('Central African Republic', 'South Africa', 'Sri Lanka'),
 ('Libya', 'Spain', 'Uruguay'),
 ('Ghana', 'Philippines', 'Sierra Leone'),
 ('Estonia', 'France', 'Saint Lucia'),
 ('Hungary', 'Panama', 'Puerto Rico'),
 ('Germany', 'Morocco', 'Tajikistan'),
 ('Cyprus', 'Guatemala', 'Portugal'),
 ('Cameroon', 'Canada', 'Swaziland'),
 ('Canada', 'India', 'Montserrat'),
 ('Cyprus', 'Italy', 'Liechtenstein'),
 ('Australia', 'Botswana', 'Nauru'),
 ('Bangladesh', 'Saint Lucia', 'United Kingdom'),
 ('Dominica', 'Gibraltar', 'United Kingdom'),
 ('Ireland', 'Mongolia', 'New Zealand'),
 ('Germany', 'Haiti', 'Serbia'),
 ('Colombia', 'Serbia', 'Sweden'),
 ('Cabo Verde', 'France', 'Jordan'),
 ('Central African Republic', 'Italy', 'Latvia'),
 ('Congo', 'Germany', 'New Zealand'),
 ('Brazil', 'New Zealand', 'Suriname'),
 ('France', 'Guatemala', 'Zambia'),
 ('Bangladesh', 'Nicaragua', 'United States of America'),
 ('France', 'Georgia', 'Haiti'),
 ('France', 'Japan', 'Mali'),
 ('Bangladesh', 'Brazil', 'Guatemala'),
 ('Austria', 'Dominican Republic', 'Libya'),
 ('Colombia', 'Egypt', 'Haiti'),
 ('Antigua and Barbuda', 'Estonia', 'France'),
 ('Canada', 'Ecuador', 'Sri Lanka'),
 ('Australia', 'Cayman Islands', 'Jordan'),
 ('Australia', 'Israel', "Lao People's Democratic Republic"),
 ('Gabon', 'Germany', 'Mexico'),
 ('Antigua and Barbuda', 'Australia', 'Sri Lanka'),
 ('Maldives', 'Oman', 'United Kingdom'),
 ('Dominica', 'France', 'Morocco'),
 ('Cambodia', 'Germany', 'Venezuela (Bolivarian Republic of)'),
 ('Brazil', 'Ireland', 'San Marino'),
 ('Bermuda', 'United Arab Emirates', 'United Kingdom'),
 ('Congo', 'Liberia', 'Mali'),
 ('Canada', 'Dominica', 'El Salvador'),
 ('Guinea-Bissau', 'Portugal', 'Venezuela (Bolivarian Republic of)'),
 ('Canada', 'Egypt', 'Saint Lucia'),
 ('New Caledonia', 'Papua New Guinea', 'Vanuatu'),
 ('Anguilla', 'Dominican Republic', 'Finland'),
 ('Belarus', 'Bosnia and Herzegovina', 'Italy'),
 ('Belarus', 'Belgium', 'Mauritania'),
 ('Central African Republic', 'Malta', 'Switzerland'),
 ('Montserrat', 'Papua New Guinea', 'United Kingdom'),
 ('Chile', 'Luxembourg', 'Nicaragua'),
 ('Comoros', 'France', 'Serbia'),
 ('Angola', 'Cabo Verde', 'South Africa'),
 ('Kyrgyzstan', 'New Zealand', 'Republic of Korea'),
 ('Cambodia', 'Netherlands', 'Nicaragua'),
 ('South Africa', 'Switzerland', 'Tajikistan'),
 ('Andorra', 'United Kingdom', 'Vanuatu'),
 ('Canada', 'Samoa', 'Spain'),
 ('Germany', 'Libya', 'Lithuania'),
 ('Lithuania', 'Samoa', 'United States of America'),
 ('Bahamas', 'France', 'Puerto Rico'),
 ('France', 'Mali', 'Mexico'),
 ('Canada', 'Haiti', 'India'),
 ('Belarus', 'Canada', 'Namibia'),
 ('Brazil', 'Ecuador', 'Lithuania'),
 ('Australia', 'Estonia', 'Solomon Islands'),
 ('Cabo Verde', 'Samoa', 'United Kingdom'),
 ('Australia', 'Republic of Korea', 'Tokelau'),
 ('Cameroon', 'Lithuania', 'United Kingdom'),
 ('Egypt', 'United Arab Emirates', 'Zambia'),
 ('Honduras', 'Jamaica', 'Trinidad and Tobago'),
 ('Central African Republic', 'Egypt', 'Togo'),
 ('Canada', 'Slovenia', 'Togo'),
 ('France', 'Georgia', 'Mexico'),
 ('Japan', 'Netherlands', 'Suriname'),
 ('India', 'Kiribati', 'United Kingdom'),
```

```
('Micronesia (Fed. States of)', 'Switzerland', 'United States of America'),
('Croatia', 'Singapore', 'United States of America'),
('Belarus', 'Panama', 'United Kingdom'),
('Mozambique', 'South Africa', 'United States of America'),
('Chile', 'Puerto Rico', 'Romania'),
('Ghana', 'Poland', 'Togo'),
('Cook Islands', 'Samoa', 'Tuvalu'),
('Dominican Republic', 'Germany', 'Lithuania'),
('Central African Republic', 'France', 'India'),
('China, Hong Kong SAR', 'Solomon Islands', 'United Kingdom'),
('Cameroon', 'United Republic of Tanzania', 'United States of America'),
('Cambodia', 'Peru', 'Russian Federation'),
('Canada', 'Uganda', 'Viet Nam'),
('Colombia', 'Poland', 'TFYR Macedonia'),
('Central African Republic', 'Ireland', 'Mongolia'),
('Cura√ßao', 'Netherlands', 'Philippines'),
('France', 'Mongolia', 'Qatar'),
('Canada', 'Guinea', 'Paraguay'),
('Brazil', 'Malta', 'Venezuela (Bolivarian Republic of)'),
('France', 'Israel', 'Republic of Korea'),
('Costa Rica', 'France', 'Zambia'),
('Italy', 'Liberia', 'TFYR Macedonia'),
('Mauritius', 'Montenegro', 'Switzerland'),
('Colombia', 'Cura√ßao', 'Israel'),
('Mexico', 'Montenegro', 'United Kingdom'),
('Israel', 'Nicaragua', 'Venezuela (Bolivarian Republic of)'),
('Honduras', 'Poland', 'Spain'),
('Australia', 'Falkland Islands (Malvinas)', 'Finland'),
('Germany', 'Paraguay', 'Serbia'),
('British Virgin Islands', 'Denmark', 'Dominican Republic'),
('Cameroon', 'Panama', 'United Kingdom'),
('Benin', 'France', 'San Marino'),
('Canada', 'South Sudan', 'Uganda'),
('Australia', 'China, Hong Kong SAR', 'Tokelau'),
('Botswana', 'Canada', 'Colombia'),
('Bosnia and Herzegovina', 'Italy', 'San Marino'),
('Colombia', 'France', 'Ghana'),
('Canada', 'Slovakia', 'Turks and Caicos Islands'),
('Egypt', 'Maldives', 'State of Palestine'),
('Cura√ßao', 'Netherlands', 'Norway'),
('Belgium', 'Mauritania', 'Namibia'),
('Japan', 'New Zealand', 'Tokelau'),
('Cuba', 'Mali', 'Russian Federation'),
('France', 'Greenland', 'Madagascar'),
('Australia', 'Brunei Darussalam', 'Tokelau'),
('Bahrain', 'Philippines', 'Russian Federation'),
('Colombia', 'Mongolia', 'Poland'),
('Philippines', 'Timor-Leste', 'Turkey'),
('El Salvador', 'Sweden', 'Turkey'),
('Cabo Verde', 'Guinea-Bissau', 'Mauritania'),
('Cura√ßao', 'Czechia', 'Netherlands'),
('Guatemala', 'Portugal', 'Slovakia'),
('Ghana', 'Philippines', 'Timor-Leste'),
('Ireland', 'Liechtenstein', 'Namibia'),
('New Zealand', 'Russian Federation', 'Uzbekistan'),
('El Salvador', 'France', 'Western Sahara'),
('Cambodia', 'Luxembourg', 'Netherlands'),
('Egypt', 'Israel', 'Sudan'),
('Bahrain', 'France', 'Seychelles'),
('Albania', 'Lithuania', 'Turkey'),
('Chile', 'Guatemala', 'Slovakia'),
('Ireland', 'Japan', 'Liechtenstein'),
('Bahamas', 'Bhutan', 'France'),
('Colombia', 'South Africa', 'United Republic of Tanzania'),
('Guatemala', 'Mexico', 'Serbia'),
('Australia', 'Bulgaria', 'Niue'),
('Antigua and Barbuda', 'Bolivia (Plurinational State of)', 'France'),
('Cabo Verde', 'China, Hong Kong SAR', 'Portugal'),
('Belgium', 'Georgia', 'Morocco'),
('Cayman Islands', 'Montserrat', 'United Kingdom'),
('El Salvador', 'Ireland', 'Poland'),
('Central African Republic', 'Italy', 'Montenegro'),
('Antigua and Barbuda', 'France', 'Montenegro'),
('Chile', 'Estonia', 'Falkland Islands (Malvinas)'),
('Tunisia', 'Ukraine', 'United States of America'),
('Honduras', 'Tajikistan', 'United Kingdom'),
('Albania', 'British Virgin Islands', 'Canada'),
('Liberia', 'Philippines', 'Spain'),
('Iran (Islamic Republic of)', 'Pakistan', 'Turkmenistan'),
('Eritrea', 'France', 'Mauritius'),
('Chile', 'Croatia', 'Serbia'),
('Anguilla', 'Canada', 'Philippines'),
('Eritrea', 'New Zealand', 'United Kingdom'),
('Czechia', 'Namibia', 'Venezuela (Bolivarian Republic of)'),
('Cambodia', 'Egypt', 'Italy'),
('Bahamas', 'Canada', 'Papua New Guinea'),
('France', 'Israel', 'Seychelles'),
('Haiti', 'Hungary', 'Spain'),
('Bulgaria', 'Namibia', 'Nicaragua'),
('Colombia', 'Kyrgyzstan', 'Latvia'),
('Nicaragua', 'Romania', 'Russian Federation'),
('Australia', 'Ghana', 'Guatemala'),
('Barbados', 'Cambodia', 'Canada'),
('Botswana', 'Lithuania', 'Norway'),
('Cambodia', 'Faeroe Islands', 'United Kingdom'),
('Latvia', 'United States of America', 'Uruguay'),
('Libya', 'Netherlands', 'Uganda'),
('Bahrain', 'Netherlands', 'Russian Federation'),
('Czechia', 'Jordan', 'Oman'),
('Mauritania', 'Tunisia', 'Western Sahara'),
('Mauritania', 'Mexico', 'Spain'),
('Antigua and Barbuda', 'Guyana', 'United States Virgin Islands'),
('Bhutan', 'Slovakia', 'Switzerland'),
('Austria', 'Central African Republic', 'Ecuador'),
('Bolivia (Plurinational State of)', 'Brazil', 'Suriname'),
('Dominican Republic', 'Jordan', 'Spain'),
('Malawi', 'Panama', 'South Africa'),
('Czechia', 'Peru', 'Puerto Rico'),
('Bosnia and Herzegovina', 'Germany', 'Guatemala'),
('Brunei Darussalam', 'Croatia', 'Netherlands'),
('Japan', 'Netherlands', 'United Arab Emirates'),
('Guyana', 'Samoa', 'United States of America'),
('Belgium', 'Egypt', 'Nicaragua'),
('Canada', 'Gabon', 'Puerto Rico'),
('China, Hong Kong SAR', 'Serbia', 'United States of America'),
('Argentina', 'Germany', 'Montenegro'),
('Bhutan', 'Denmark', 'Faeroe Islands'),
('Croatia', 'Greece', 'Sri Lanka'),
('Canada', 'Mexico', 'Namibia'),
('Brazil', 'Liechtenstein', 'Namibia'),
('Central African Republic', 'Egypt', 'Norway'),
('Liberia', 'Lithuania', 'Switzerland'),
('Cyprus', 'Guyana', 'United States of America'),
('Belarus', 'Norway', 'Philippines'),
('Japan', 'Tunisia', 'United States of America'),
('Mali', 'Mauritius', 'Russian Federation'),
('Belgium', 'Mauritius', 'Panama'),
('Belarus', 'Ireland', 'Mexico'),
('Jamaica', 'Maldives', 'United Kingdom'),
('Argentina', 'Guinea', 'Uruguay'),
('Canada', 'Serbia', 'Uruguay'),
('Italy', 'Liechtenstein', 'Maldives'),
('Japan', 'Kyrgyzstan', 'Republic of Korea'),
('Central African Republic', 'Haiti', 'Italy'),
('Canada', "Dem. People's Republic of Korea", 'Guyana'),
('Austria', 'Belarus', 'Ecuador'),
('Cuba', 'France', 'Zambia'),
('Cambodia', 'France', 'Greenland'),
('Ghana', 'Poland', 'Republic of Moldova'),
('Faeroe Islands', 'Iceland', 'Switzerland'),
('Argentina', 'Cabo Verde', 'Latvia'),
('Lesotho', 'Mauritius', 'South Africa'),
('China, Hong Kong SAR', 'Guatemala', 'United States of America'),
('Brazil', 'Lesotho', 'South Africa'),
('Bahrain', 'Egypt', 'Guatemala'),
('Guatemala', 'Luxembourg', 'Peru'),
('Australia', 'Czechia', 'Solomon Islands'),
('Brazil', 'Cabo Verde', 'Serbia'),
('Mozambique', 'Sierra Leone', 'South Africa'),
('Bulgaria', 'Greenland', 'Latvia'),
('Burkina Faso', 'Cameroon', 'Niger'),
('Italy', 'Montenegro', 'San Marino'),
('France', 'Haiti', 'Zambia'),
('Slovakia', 'Tajikistan', 'Turkey'),
('Colombia', 'Mozambique', 'South Africa'),
('France', 'Lithuania', 'New Zealand'),
('Algeria', 'Congo', 'Germany'),
('Cambodia', 'Lebanon', 'United States of America'),
('Faeroe Islands', 'Norway', 'Slovenia'),
('Azerbaijan', 'Jordan', 'Turkey'),
('Belarus', 'Norway', 'Sri Lanka'),
('Australia', 'Bahamas', 'Malta'),
('Ecuador', 'Liberia', 'Netherlands'),
('Botswana', 'Libya', 'South Africa'),
('Azerbaijan', 'Ghana', 'Russian Federation'),
('Australia', 'Croatia', 'Isle of Man'),
('Botswana', 'Italy', 'Tunisia'),
('South Africa', 'Sri Lanka', 'Swaziland'),
('Colombia', 'Italy', 'Republic of Moldova'),
('Italy', 'Liberia', 'Panama'),
('Caribbean Netherlands', 'Saint Lucia', 'Saint Vincent and the Grenadines'),
('Botswana', 'Malta', 'South Africa'),
('Cameroon', 'Cuba', 'France'),
('Aruba', 'Cyprus', 'Venezuela (Bolivarian Republic of)'),
('Lithuania', 'Nicaragua', 'Russian Federation'),
('Canada', "Dem. People's Republic of Korea", 'Mauritius'),
('Canada', 'Ecuador', 'Mauritania'),
('Bermuda', 'Lithuania', 'United Kingdom'),
('Cambodia', 'Netherlands', 'Uganda'),
('Canada', 'Samoa', 'Thailand'),
('Cambodia', 'France', 'Liechtenstein'),
('Cuba', 'Dominican Republic', 'Turks and Caicos Islands'),
('Australia',
 "Lao People's Democratic Republic",
 'Venezuela (Bolivarian Republic of)'),
('Congo', 'France', 'Iceland'),
('Bahrain', 'Ghana', 'United Kingdom'),
('Croatia', 'New Zealand', 'Samoa'),
('Bulgaria', 'France', 'Seychelles'),
('Austria', 'Colombia', 'Tajikistan'),
('Antigua and Barbuda', 'Australia', 'Guatemala'),
('Republic of Korea', 'South Africa', 'Swaziland'),
('Antigua and Barbuda', 'Egypt', 'Sweden'),
('Bosnia and Herzegovina', 'France', 'Philippines'),
('Canada', 'Estonia', 'Papua New Guinea'),
('Austria', 'Portugal', 'Timor-Leste'),
...}
```

```
In [ ]:
```