Class Visualizing Migration Network and Plotting Degree Distibutions

Quantifying Global International Migration Flows

Guy J. Abel[*], **Nikola Sander**[*,†]

+ See all authors and affiliations

Science 28 Mar 2014:
Vol. 343, Issue 6178, pp. 1520-1522
DOI: 10.1126/science.1248676

Monitoring Migration

Migrant "stock" data—the number of people living in a country other than the one in which they were born—are frequently used to understand contemporary trends in international migration, but the data are severely limited. **Abel and Sander** (p. 1520) present a set of global bilateral migration flows estimated from sequential stock data in 5-year intervals. The percentage of the world population moving over 5-year periods has not shown dramatic changes between 1995 and 2010. People from individual African countries tended to move within the continent, whereas people from Europe tended to move to very diverse locations.

Abstract

Widely available data on the number of people living outside of their country of birth do not adequately capture contemporary intensities and patterns of global migration flows. We present data on bilateral flows between 196 countries from 1990 through 2010 that provide a comprehensive view of international migration flows. Our data suggest a stable intensity of global 5-year migration flows at ~0.6% of world population since 1995. In addition, the results aid the interpretation of trends and patterns of migration flows to and from individual countries by placing them in a regional or global context. We estimate the largest movements to occur between South and West Asia, from Latin to North America, and within Africa.

Supplementary Materials

Quantifying Global International Migration Flows

Guy J. Abel, Nikola Sander

Materials/Methods, Supplementary Text, Tables, Figures, and/or References

Download Supplement

- Materials and Methods
- Tables S1 to S5
- Full References List

Database S1

- Bilateral flow estimates by region, 2005-10
- This is a 15*15 matrix stored as an excel file. Rows correspond to origins, columns to destinations.

Database S2

- Bilateral flow estimates by country, 1990-95 to 2005-10
- This is a 196*196 matrix stored as an excel file. Rows correspond to origins, columns to destinations. Countries a indicated by their iso-3 code.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	Aruba	Afghanistan	Angola	Albania	United Arab	Argentina	Armenia	Australia	Austria	Azerbaijan	Burundi	Belgium	Benin	Burkina Fasc	Bangladesh	Bulgaria	Bahrain	Bahamas	Bosnia and
Aruba	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
Afghanistan	0	0	0	0	2094	0	0	8885	2175	35	0	1068	0	0	0	136	280	0	
Angola	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Albania	0	0	0	0	0	0	0	12	0	0	0	432	0	0	0	329	0	0	
United Arab	0	0	0	0	0	0	0	7842	209	0	0	0	0	0	0	104	11623	0	
Argentina	6	0	98	24	55	0	0	3653	995	17	5	1210	0	0	0	466	16	44	
Armenia	0	0	0	1	7	0	0	11	21	41153	0	337	0	0	0	777	1	0	
Australia	0	0	4	770	1	0	0	0	3214	54	0	1525	0	0	0	653	0	0	
Austria	0	0	1	96	0	0	0	299	0	8	53	256	0	0	0	47	0	0	
Azerbaijan	0	0	0	0	0	0	0	0	44	0	0	7	0	0	0	4	0	0	
Burundi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Belgium	0	0	4	3	10	0	0	18	262	1	356	0	0	0	0	84	1	0	
Benin	0	0	0	0	0	0	0	8	5	0	0	21	0	0	0	5	0	0	
Burkina Fasc	0	0	0	0	0	4	0	87	95	0	0	162	5487	0	0	12	0	0	
Bangladesh	9	0	1710	2	646729	70	0	28250	1910	5	19	1199	0	0	0	681	87794	8	
Bulgaria	0	0	0	0	0	0	0	0	0	0	0	3927	0	0	0	0	0	1	
Bahrain	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bahamas	0	0	0	0	2	0	0	130	13	0	0	4	0	0	0	4	0	0	
Bosnia and H	0	0	0	1534	1	0	0	1100	694	0	0	174	0	0	0	96	0	0	
Belarus	0	0	0	0	0	0	0	0	83	763	0	336	0	0	0	231	0	0	
Belize	0	0	0	0	0	0	0	14	9	0	0	0	0	0	0	1	0	2	
Bolivia	0	0	0	0	0	56615	0	116	98	0	1	0	0	0	0	13	0	0	
Brazil	22	0	3026	9	61	6715	0	9119	2092	1	2	5398	0	0	0	291	7	23	
Barbados	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
Brunei	2	0	0	0	1597	2	0	10236	50	0	0	16	0	0	496	1	199	3	
Bhutan	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Botswana	0	0	5	0	0	0	0	343	6	0	267	0	0	0	0	14	0	6	
Central Afric	0	0	11	0	0	0	0	10	3	0	850	0	0	0	0	0	0	0	
Canada	0	0	20	35	9	0	0	2403	2162	35	2121	2586	0	0	0	327	62	1	
Switzerland	0	0	1	5	3	0	0	768	2427	0	45	463	0	0	0	96	0	0	
Channel Islar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Chile	0	0	5	1	3	4243	0	4223	361	1	6	173	0	0	0	20	1	9	
China	191	0	0	0	0	2736	0	150947	4402	151	0	4474	0	0	0	465	0	20	
Ivory Coast	0	0	159	0	25	1	0	992	406	0	0	2782	13631	119154	1	28	3	1	
Cameroon	0	0	0	0	0	0	0	123	183	0	0	4606	0	0	0	5	0	0	
Democratic F	0	0	710	0	785	0	0	305	4	0	16780	4	3	0	0	1	92	0	
Republic of (	0	0	159	0	0	0	0	149	7	0	593	13	10	0	0	1	0	0	

[illegible]

```
,Aruba,Afghanistan,Angola,Albania,United Arab Emirates,Argentina,Armenia,Australia,Austria,Azerbaijan,Burundi,Belgium,Benin,Burkina Faso,Bangladesh,Bulgaria,Bahrain,Bahamas,Bosnia and Herzegovi
na,Belarus,Belize,Bolivia,Brazil,Barbados,Brunei,Bhutan,Botswana,Central African Republic,Canada,Switzerland,Channel Islands,Chile,China,Ivory Coast,Cameroon,Democratic Republic of the Congo,Re
public of Congo,Colombia,Comoros,Cape Verde,Costa Rica,Cuba,Cyprus,Czech Republic,Germany,Djibouti,Denmark,Dominican Republic,Algeria,Ecuador,Egypt,Eritrea,Western Sahara,Spain,Estonia,Ethiopia
,Finland,Fiji,France,Micronesia,Gabon,United Kingdom,Georgia,Ghana,Guinea,Guadeloupe,Gambia,Guinea-Bissau,Equatorial Guinea,Greece,Grenada,Guatemala,French Guiana,Guam,Guyana,Hong Kong,Honduras
,Croatia,Haiti,Hungary,Indonesia,India,Ireland,Iran,Iraq,Iceland,Israel,Italy,Jamaica,Jordan,Japan,Kazakhstan,Kenya,Kyrgyzstan,Cambodia,South Korea,Kuwait,Laos,Lebanon,Liberia,Libya,Saint Lucia
,Sri Lanka,Lesotho,Lithuania,Luxembourg,Latvia,Macao,Morocco,Moldova,Madagascar,Maldives,Mexico,Macedonia,Mali,Malta,Myanmar,Montenegro,Mongolia,Mozambique,Mauritania,Martinique,Mauritius,Malaw
i,Malaysia,Mayotte,Namibia,New Caledonia,Niger,Nigeria,Nicaragua,Netherlands,Norway,Nepal,New Zealand,Oman,Pakistan,Panama,Peru,Philippines,Papua New Guinea,Poland,Puerto Rico,North Korea,Portu
gal,Paraguay,Palestine,French Polynesia,Qatar,Reunion,Romania,Russia,Rwanda,Saudi Arabia,Sudan,Senegal,Singapore,Solomon Islands,Sierra Leone,El Salvador,Somalia,Serbia,South Sudan,Sao Tome and
 Principe,Suriname,Slovakia,Slovenia,Sweden,Swaziland,Syria,Chad,Togo,Thailand,Tajikistan,Turkmenistan,East Timor,Tonga,Trinidad and Tobago,Tunisia,Turkey,Tanzania,Uganda,Ukraine,Uruguay,United
 States,Uzbekistan,Saint Vincent and the Grenadines,Venezuela,Virgin Islands,Vietnam,Vanuatu,Samoa,Yemen,South Africa,Zambia,Zimbabwe
Aruba,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,14,0,0,1,0,0,0,0,0,0,0,0,13,0,0,0,42,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,
0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,87,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10
1,0,0,0,0,0,0,0,0
Afghanistan,0,0,0,0,2094,0,0,8885,2175,35,0,1068,0,0,0,136,280,0,0,133,0,1,26,0,0,0,0,0,12270,1247,0,3,0,0,0,0,0,36,0,0,1,0,3,326,32852,0,4478,1,2,33,34,0,0,0,3,0,1379,0,1091,0,0,13444,9,0,0,0,
0,0,0,62,0,0,0,0,0,0,0,0,0,316,0,0,44,266595,18,3,810,1782,0,4,0,567,0,12,0,7,104,0,0,0,37,0,0,0,10,0,17,0,0,63,0,0,2,0,0,1,0,0,0,0,1,0,0,0,145,0,0,0,0,0,3,8894,4701,0,896,0,0,3,0,210,0,62,0,0,
5,0,0,0,550,0,3,5731,0,796,0,0,0,0,0,0,0,0,0,0,63,4,6388,0,566,0,0,0,0,20,0,0,0,0,4119,0,0,315,0,13781,192,0,5,0,0,0,0,0,13,0,0
Angola,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0
Albania,0,0,0,0,0,0,0,12,0,0,0,432,0,0,0,329,0,0,0,0,0,0,0,0,0,0,0,38,0,0,0,0,0,0,0,0,0,0,0,13,117,5495,0,41,0,0,10,0,0,0,146,0,0,46,0,196,0,0,4402,0,0,0,0,0,0,0,8912,0,0,0,0,0,0,0,0,0,21
,0,0,0,0,0,4,0,48150,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,6986,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,60,75,0,0,0,0,0,0,0,0,0,40,0,0,0,0,0,0,0,0,0,194,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,4,240,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
United Arab Emirates,0,0,0,0,0,0,0,7842,209,0,0,0,0,0,0,104,11623,0,0,0,0,3,47,0,0,0,0,0,15701,507,0,7,0,0,0,0,0,57,0,0,0,2,190,8,0,0,463,0,896,9,1381,0,0,0,0,0,126,0,35392,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,281,0,12104,370,0,253,3,0,437,0,238,0,0,0,0,0,0,25166,0,0,0,2817,0,0,0,60,0,17,0,0,0,0,0,0,10,0,0,19,0,0,0,0,185,0,0,0,0,0,0,0,0,0,3,789,389,0,1166,6181,0,2,8,431,0,235,0,0,2,
0,13960,0,24049,0,173,564,0,0,0,0,0,0,0,0,0,0,4924,0,0,125,18,1101,0,0,0,0,0,0,0,0,0,0,0,49,0,0,0,0,11196,0,0,10,0,0,0,0,0,0,209,0,0
Argentina,6,0,98,24,55,0,0,3653,995,17,5,1210,0,0,0,466,16,44,0,51,2,13192,2466,2,1,0,31,17,6996,3546,20,10015,23,0,0,1,17,2226,0,35,143,4,111,2422,5863,0,805,602,103,7313,78,0,0,136370,16,0,49
8,0,3448,0,0,3696,0,2,0,1,0,0,269,763,0,176,0,0,2,7,22,55,254,651,8,0,689,3,2,137,18147,91127,3,8,678,921,0,1,0,32,0,0,2,0,3,1,0,0,108,321,82,0,0,7,8,0,3860,8,0,14,0,17,2,0,56,0,4,0,0,0,0,12,0,
0,14,1097,998,0,611,0,0,665,36,260,0,1820,0,1,1392,22126,2,1,14,0,3107,1325,5,3,0,0,18,0,0,30,0,45,9,0,16,456,308,1658,0,10,0,0,130,0,1,0,0,1,12,820,40,31,343,3682,28117,10,0,3837,0,24,0,0,0,29
07,0,0
Armenia,0,0,0,1,7,0,0,11,21,41153,0,337,0,0,0,777,1,0,0,77,0,2,5,0,0,0,0,0,53,127,0,0,3,0,0,0,3,1,0,1,0,0,60,633,172,0,54,0,4,12,0,0,0,360,19,0,30,0,1158,0,0,10,42,0,0,0,0,0,0,174,0,0,0,0,0,0,0
,0,0,40,0,0,0,2544,3,1,12,546,0,0,0,12905,0,59,0,1,0,0,0,0,0,0,0,19,0,1,0,0,299,0,0,0,4,0,1,0,1,6,0,0,0,0,0,0,0,0,0,0,0,159,140,0,3,0,0,0,0,0,0,137,0,0,44,0,0,0,2,0,41,10551,0,0,0,0,0,0,0,0
,0,0,0,0,0,13,4,698,0,0,0,0,0,0,111,0,0,0,0,417,0,0,2867,0,4,416,0,0,0,0,0,0,0,4,0,0
Australia,0,0,4,770,1,0,0,0,3214,54,0,1525,0,0,0,653,0,0,0,118,0,63,206,8,0,1,105,9,3577,2548,1,47,1554,0,0,0,7,99,0,3,2,0,1764,5307,13840,0,1415,5,140,553,103,0,3,9483,37,0,883,293,2254,0,0,22
546,0,5,0,0,0,0,0,13132,0,11,0,0,0,407,0,580,0,2611,641,0,911,1,0,322,1939,47436,0,0,1058,1667,0,1,0,1299,0,2,0,0,0,5,0,0,138,247,136,0,0,22,9,0,53,258,0,334,0,405,111,0,1,0,31,0,0,0,0,54,0,0,0
,1559,2486,0,9918,0,0,84,0,450,0,1400,0,97,498,8,0,1,0,0,1939,2480,11,0,0,0,1756,0,0,2,0,6544,1137,0,0,1195,908,2314,21,0,0,0,0,0,7,0,0,0,7,6846,38,18,414,0,308,42,0,29,0,30,155,0,0,1729,0,0
Austria,0,0,1,96,0,0,0,299,0,8,53,256,0,0,0,47,0,0,0,0,0,2,10,0,0,0,0,0,118,677,0,1,0,0,0,11,0,1,0,0,0,0,15,4765,9500,0,85,0,9,11,5,0,0,1221,0,0,41,0,78,0,0,429,0,0,0,0,0,0,26,0,0,0,0,0,0,0,4
9,0,1669,0,0,25,1,0,13,117,1600,0,0,0,292,1,3,0,0,0,0,0,39,0,0,0,0,8,65,16,0,0,4,0,0,0,7,0,4,0,42,1,0,0,0,0,0,0,0,0,0,0,0,0,71,142,0,5,0,0,1,0,0,0,414,0,0,27,0,0,0,1,0,160,162,4,0,0,0,0,0,0,0,0
,161,190,0,0,3187,507,154,0,0,0,0,0,0,3,0,0,0,0,746,0,15,1,0,0,18,0,2,0,1,0,0,0,22,0,0
Azerbaijan,0,0,0,0,0,0,0,0,44,0,0,7,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,48,0,0,0,0,0,0,0,0,0,0,0,3,13,1,0,3,0,1,1,1,0,0,58,0,0,3,0,21,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,0,0,2,5,0,0,10,1
41,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,8,0,1,0,0,0,0,0,0,22,0,0,1,0,0,0,0,0,4,249,0,0,0,0,0,0,0,0,0,1,0,0,0,2,2,10,0,0,0,0,0,0,0,0,0,0,162,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Burundi,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0
Belgium,0,0,4,3,10,0,0,18,262,1,356,0,0,0,0,84,1,0,4,0,0,6,0,0,0,0,0,0,8,878,0,0,0,0,0,0,6,0,0,0,0,27,118,3004,0,152,0,22,42,7,0,0,1384,0,0,78,0,1650,0,0,390,0,0,0,0,0,0,267,0,0,0,0,0,0,0,6
,0,215,0,0,120,1,2,6,594,9910,0,0,0,75,4,1,0,0,0,0,0,0,0,0,0,24,1654,27,0,0,0,2,0,0,17,0,5,0,1,0,0,3,0,0,0,0,0,0,0,0,0,36,136,0,8,0,0,6,0,0,0,1196,0,0,58,0,0,0,3,0,310,12,67,0,0,0,0,0,0,0,0
:
```

In [32]:
```
# run this from a terminal, to install the packet:
#pip install geopy
import geopy
from geopy.geocoders import Nominatim
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import csv
from os.path import join
#this will allow the plot to be inline in the browser
%matplotlib inline
```

In [33]:
```
lines = np.loadtxt("migration.csv",dtype='str',delimiter=',')
#links = np.loadtxt("THE_LINKS.txt",dtype=[('f0', '|S5'), ('f1', '|S5'),('f2', float), ('f3', int)],usecols=(0,1,2,3))
```

In [34]:
```
len(lines)
destinations=lines[0];
```

In [35]:
```
g = nx.DiGraph()

for i in range(1,len(lines)-1):
    count=0;
    data=lines[i];
    node=data[0];
    for col in data[1:]:
        count=count+1;
        d=destinations[count];
        wij=int(col)
        g.add_edge(node,d,weight=wij)
        #print node,' ',d,' ',col
        #print col
```

In [7]:
```
from geopy.exc import GeocoderTimedOut
from geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.geocode("USA")
print location.latitude,' ',location.longitude


### Here we use it to create the dictionary with coordinates
pos = {}
for n in g.nodes():
    print n
    location = geolocator.geocode(n,timeout=30)
    pos[n] = (location.longitude,location.latitude)
    #print location.latitude," ",location.longitude
```

```
39.7837304   -100.4458825
Canada
East Timor
Sao Tome and Principe
Turkmenistan
Lithuania
Cambodia
Ethiopia
Aruba
Swaziland
Palestine
Argentina
Bolivia
Cameroon
Burkina Faso
Ghana
Saudi Arabia
Japan
Channel Islands
Cape Verde
Slovenia
Guatemala
Bosnia and Herzegovina
Kuwait
Jordan
Spain
```

Check minimum and maximun weighted in-degree and of of nodes

In [9]:
```
min_w_deg=min(dict(g.in_degree(weight='weight')).values())

max_w_deg=max(dict(g.in_degree(weight='weight')).values())
print "min= ",min_w_deg,"and max= ",max_w_deg
in_degrees=g.in_degree(weight='weight')
    # sort nodes by degree
from operator import itemgetter
in_nodes=sorted(g.in_degree(weight='weight'),key=itemgetter(1))
```

```
min=  0 and max=  6804989
```

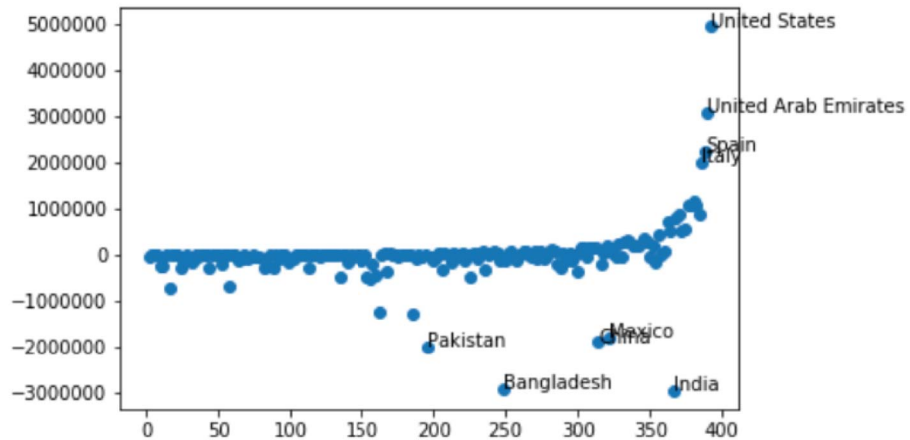
In [10]:
```
in_nodes
```

Out[10]:
```
[('East Timor', 0),
 ('Samoa', 0),
 ('Micronesia', 0),
 ('Zimbabwe', 0),
 ('Cambodia', 15),
 ('Sri Lanka', 83),
 ('Lesotho', 88),
 ('Peru', 122),
 ('Tonga', 239),
 ('Comoros', 245),
 ('Virgin Islands', 261),
 ('Tajikistan', 272),
 ('Cuba', 282),
 ('Grenada', 500),
 ('Puerto Rico', 578),
 ('Georgia', 651),
```

In [11]:
```
count=0;
z=[];
y=[];
nn=[];
for n in in_nodes:
    count=count+2;
    z.append(count);
    net=dict(g.in_degree(n,'weight')).values()[0]-dict(g.out_degree(n,'weight')).values()[0]
    y.append(net)
    nn.append(n[0]) ;
    print n[0],' ',dict(g.in_degree(n,'weight')).values()[0],' ',dict(g.out_degree(n,'weight')).values()[0]," ",net
```

```
East Timor    0    49871    -49871
Samoa    0    15735    -15735
Micronesia    0    8992    -8992
Zimbabwe    0    0    0
Cambodia    15    254784    -254769
Sri Lanka    83    249657    -249574
Lesotho    88    20282    -20194
Peru    122    725157    -725035
Tonga    239    8434    -8195
Comoros    245    10253    -10008
Virgin Islands    261    3862    -3601
Tajikistan    272    296352    -296080
Cuba    282    190423    -190141
```

In [12]:
```
fig, ax = plt.subplots()
ax.scatter(z, y)
for i, txt in enumerate(nn): #writing the names if the growth or decrease is more than 1.5M
    if(abs(y[i])>1500000):
        ax.annotate(txt, (z[i],y[i]))
```



In [13]:
```
g.out_degree('United States','weight')
```

Out[13]: 1869951

In [14]:
```
min_wout_deg=min(dict(g.out_degree(weight='weight')).values())
max_wout_deg=max(dict(g.out_degree(weight='weight')).values())
print "min= ",min_wout_deg,"and max= ",max_wout_deg
```

```
min=  0 and max=  3724386
```

In [15]:
```
g.number_of_nodes()
```

Out[15]: 196

In [16]:
```
np.median(dict(g.out_degree(weight='weight')).values())
```

Out[16]: 51930.5

In [17]:
```
np.median(dict(g.in_degree(weight='weight')).values())
```

Out[17]: 29992.5

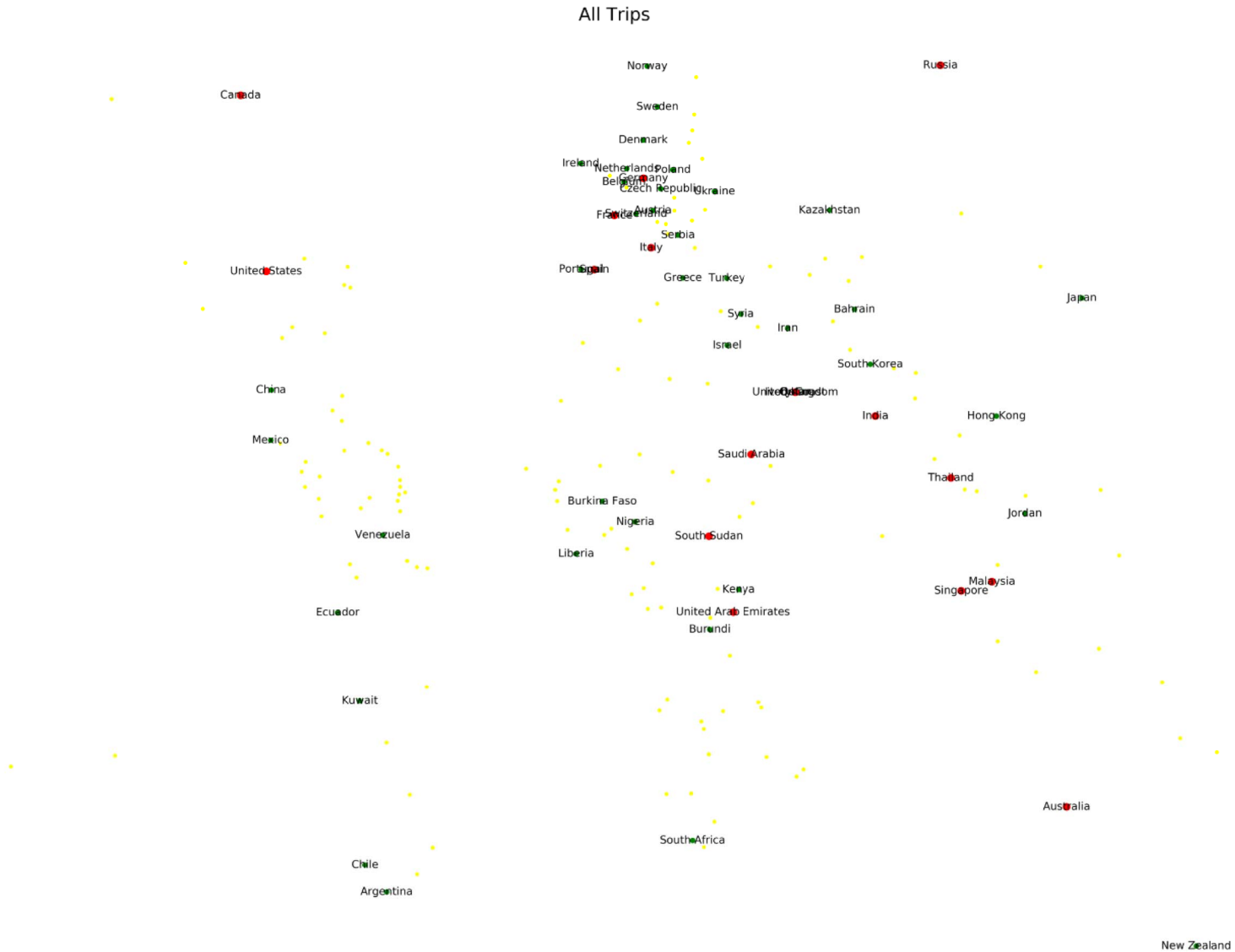
In [18]:
```
for a, b, data in sorted(g.edges(data=True), key=lambda (a, b, data): data['weight']):
    if data['weight'] > 200000:
        print('{a} {b} {w}'.format(a=a, b=b, w=data['weight']))
```

```
Indonesia United Arab Emirates 200422
China South Korea 216747
India United Kingdom 250403
China Hong Kong 251540
United States Italy 255654
Morocco Spain 265763
Afghanistan Iran 266595
El Salvador United States 268935
Pakistan Saudi Arabia 285441
Tanzania Burundi 294595
India Qatar 297570
Uzbekistan Russia 299656
Kazakhstan Russia 307495
Indonesia Malaysia 346048
Myanmar Thailand 368832
Philippines United States 373024
Malaysia Singapore 395727
Pakistan United Arab Emirates 494846
Bangladesh Saudi Arabia 523342
China United States 615536
Bangladesh India 630451
Bangladesh United Arab Emirates 646729
India United States 701242
India United Arab Emirates 1149965
Mexico United States 1957397
```

Activity: Display colors of nodes by 3 sizes respect to their incoming trips

:
```
import matplotlib.colors as colors
plt.figure(3)

degree_in=dict(g.in_degree(weight='weight')).values()

ns=[] #saves node size
nc=[] #saves node color
n_label = {}
#for n in g.nodes_iter():
 #   n_label[n] = n

for node in g.nodes():
    dn=g.in_degree(node,'weight')
    #print node
    if(dn>=500000):
        n_label[node] = node
        ns.append(4000)
        nc.append("red")
    if(dn>=100000 and dn<500000):
        ns.append(2000)
        nc.append("green")
        n_label[node] = node
    if(dn<100000):
        ns.append(1000)
        nc.append("yellow")
```

In [20]:
```
#pos=nx.spring_layout(g)
plt.figure(figsize=(200,150))
nx.draw_networkx_nodes(g, pos=pos, node_size=ns,
                       node_color = nc)
nx.draw_networkx_labels(g,pos,n_label,font_size=90,font_color='k')
plt.title('All Trips',fontsize=150)
plt.axis('off')
plt.savefig('map0', format='pdf')
```

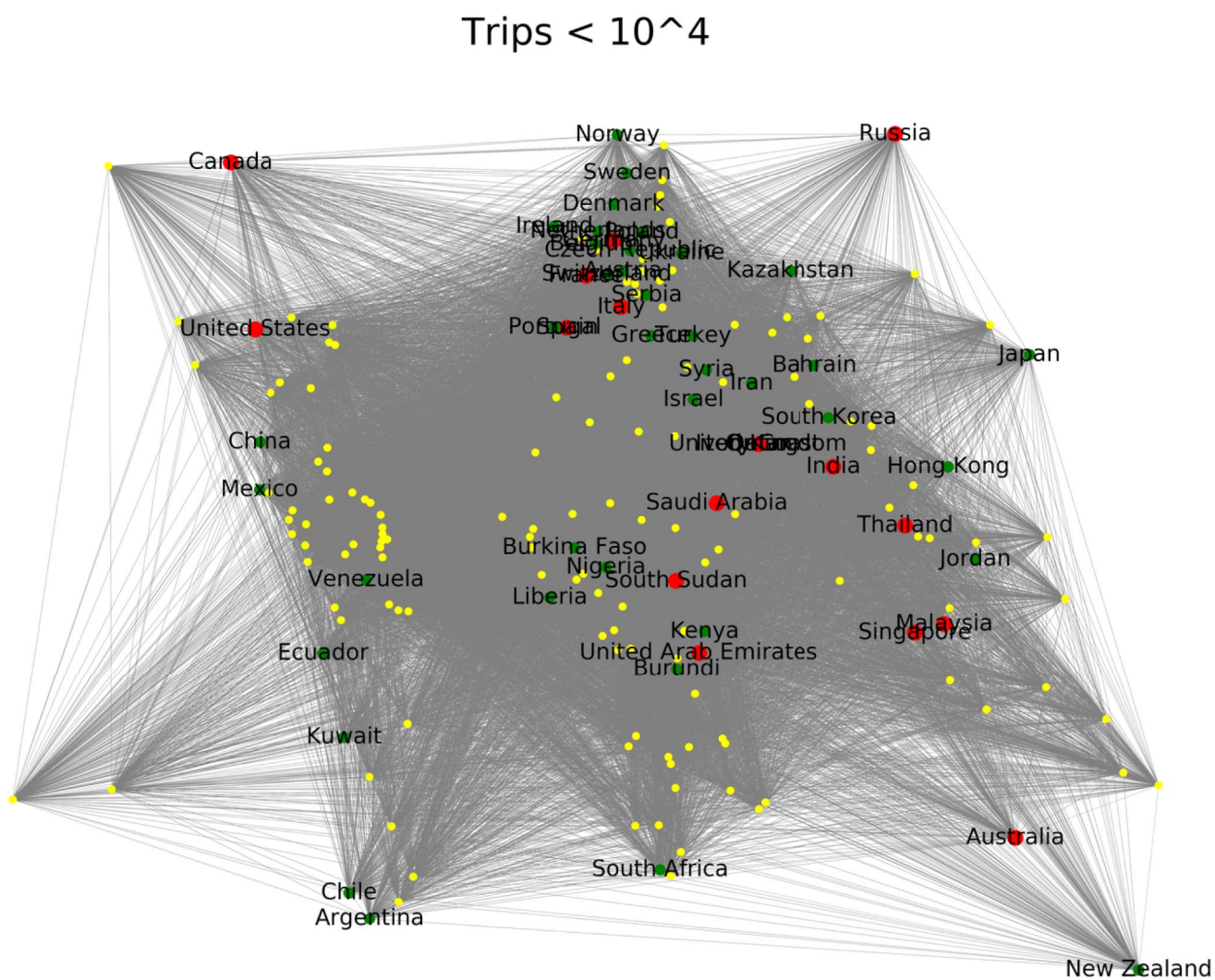


Activity: Show links only in several ranges of weight values

In [21]:
```
# Create list of edeges by color

eI = []
eII = []
eIII = []
eIV= []

for u,v in g.edges():
    wij=g.get_edge_data(u,v)['weight']
    if (wij>=500000):
       eI.append((u,v))
    if (wij>=100000 and wij<500000):
       eII.append((u,v))
    if (wij>=10000 and wij<100000):
       eIII.append((u,v))
    if (wij<10000):
       eIV.append((u,v))
```

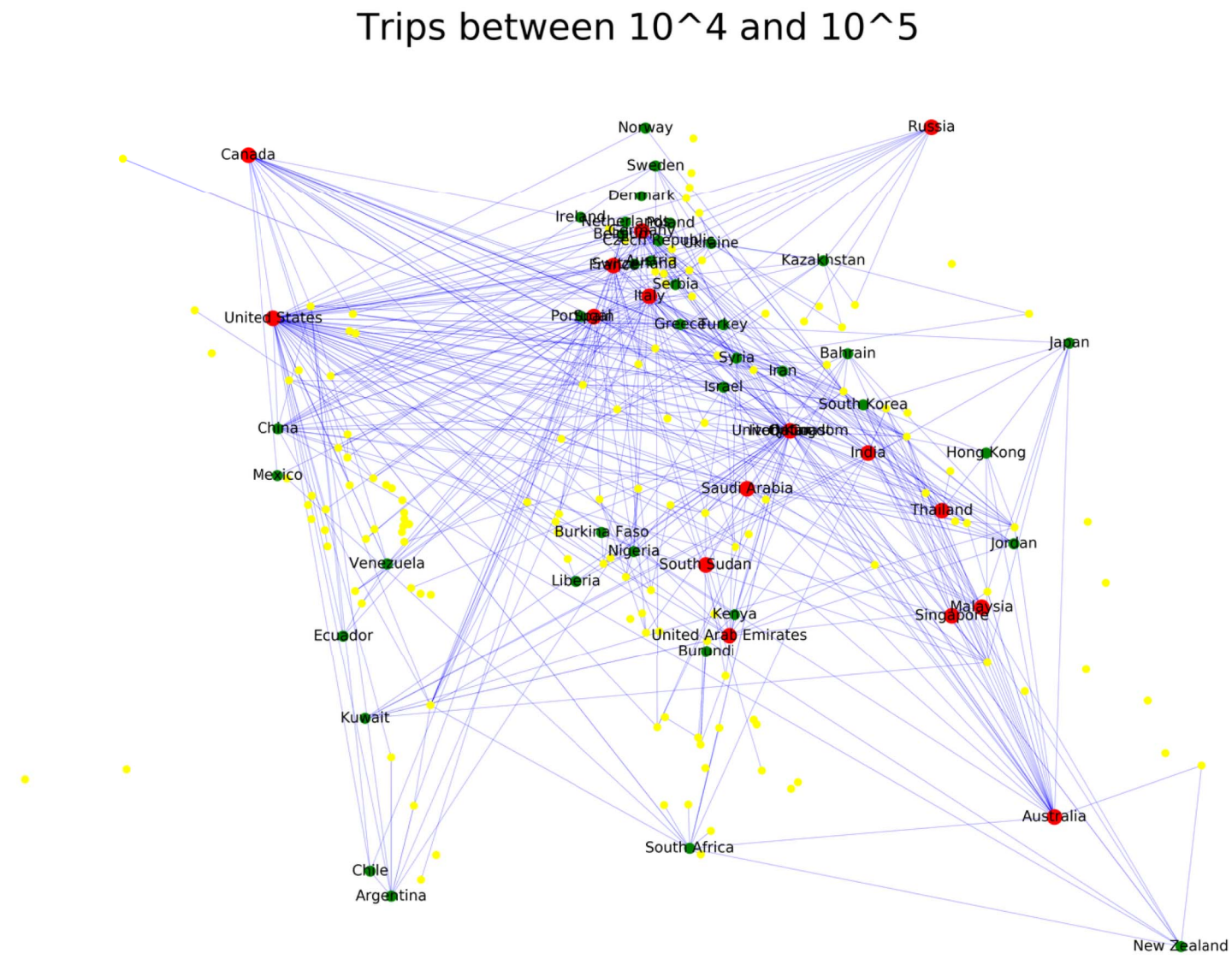
In [22]:
```
#here we show the edgelist I
plt.figure(5)
plt.figure(figsize=(100,75))
nx.draw_networkx_edges(g,pos,edgelist=eIV,width=1,alpha=0.9,edge_color='gray')
nx.draw_networkx_nodes(g,pos=pos,label=n_label,with_labels=True,node_size=ns,node_color=nc)
nx.draw_networkx_labels(g,pos,n_label,font_size=90,font_color='k')
plt.title('Trips < 10^4',fontsize=150)
plt.axis('off')
```

Out[22]:
```
(-212.39520859049998, 216.2048398905, -52.649654753499995, 75.83588525350001)

<matplotlib.figure.Figure at 0x1155cd790>
```

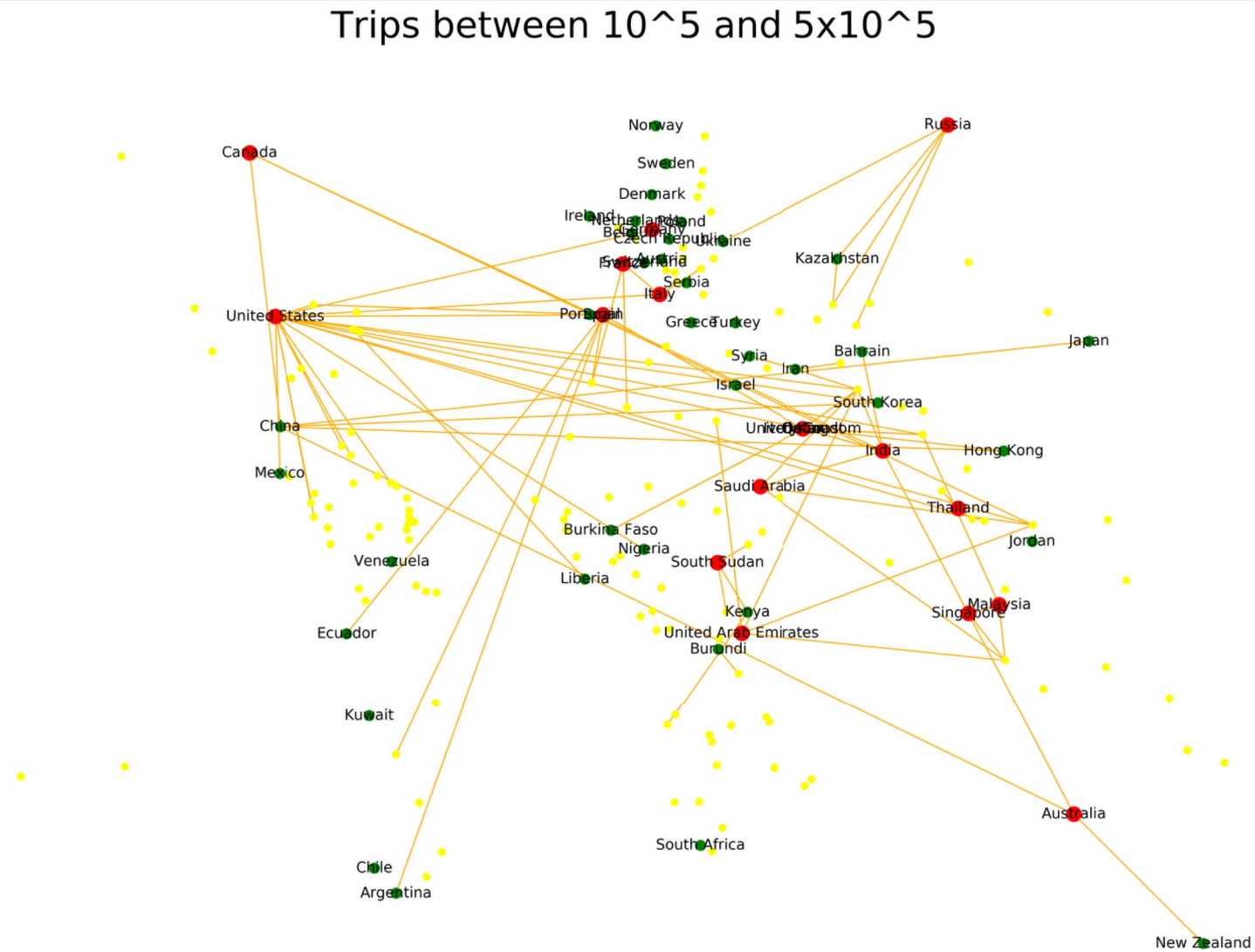


In [29]:
```
plt.figure(6)
plt.figure(figsize=(100,75))
nx.draw_networkx_edges(g, pos, edgelist = eIII, width = 1 , alpha = 0.9, edge_color='blue')
nx.draw_networkx_nodes(g,pos=pos,node_size=ns,node_color=nc,edge_color='gray')
nx.draw_networkx_labels(g,pos,n_label,font_size=60,font_color='k')
plt.title('Trips between 10^4 and 10^5',fontsize=150)
plt.axis('off')
plt.savefig('map3', format='pdf',dpi=1000)
```

```
<matplotlib.figure.Figure at 0x148613390>
```

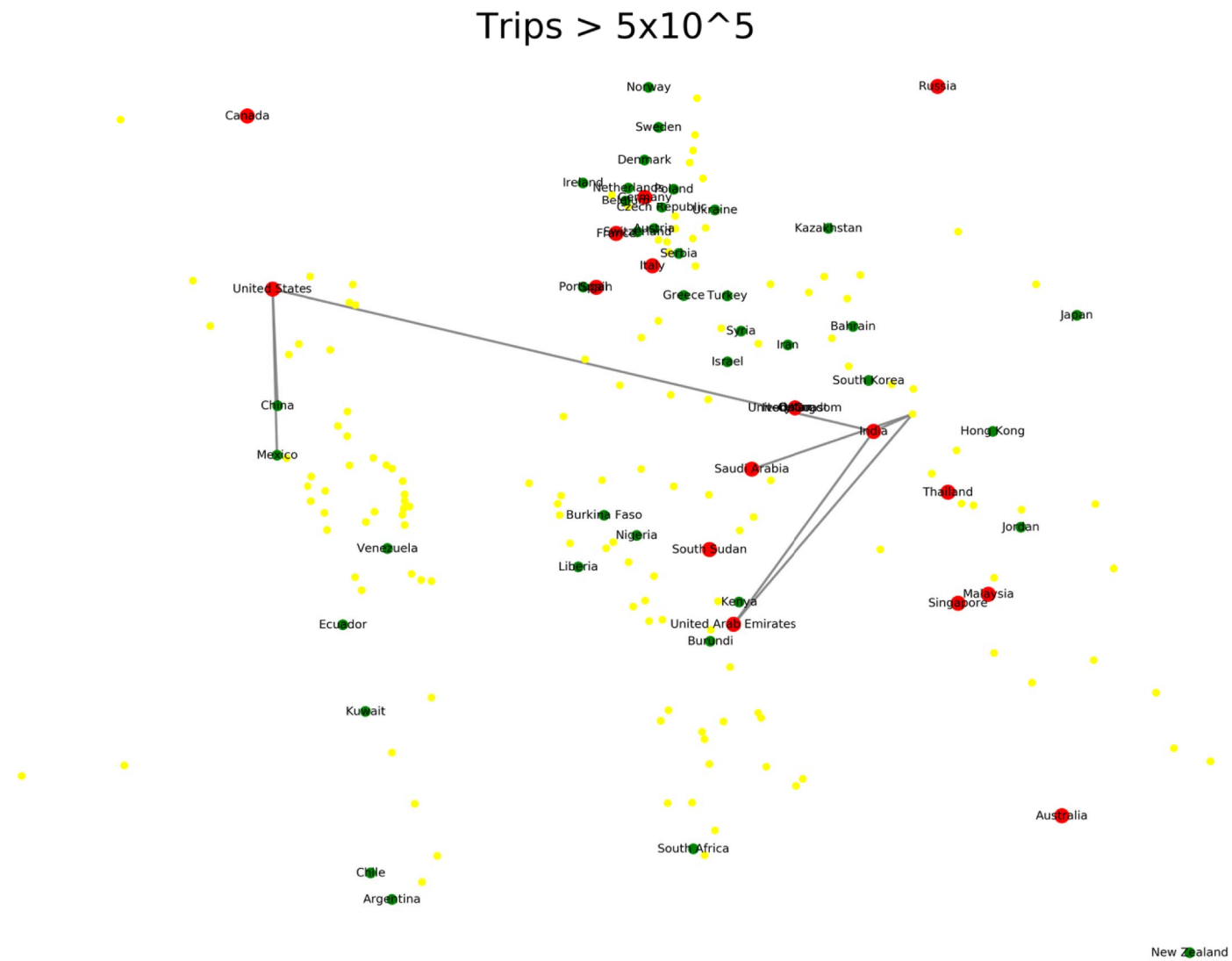


In [30]:
```
#here we show the edgelist II
plt.figure(7)
plt.figure(figsize=(100,75))
nx.draw_networkx_edges(g, pos, edgelist = eII, width = 5 , alpha = 0.9, edge_color='orange')
nx.draw_networkx_nodes(g,pos=pos,node_size=ns,node_color=nc,edge_color='gray')
nx.draw_networkx_labels(g,pos,n_label,font_size=60,font_color='k')
plt.title('Trips between 10^5 and 5x10^5',fontsize=150)
plt.axis('off')
plt.savefig('map2', format='pdf',dpi=1000)
```



In [31]:
```
#here we show the edgelist I
plt.figure(8)
plt.figure(figsize=(100,75))
nx.draw_networkx_edges(g, pos, edgelist = eI, width = 10 , alpha = 0.9, edge_color='gray')
nx.draw_networkx_nodes(g,pos=pos,node_size=ns,node_color=nc,edge_color='gray')
nx.draw_networkx_labels(g,pos,n_label,font_size=50,font_color='k')
plt.title('Trips > 5x10^5',fontsize=150)
plt.axis('off')
plt.savefig('map1', format='pdf',dpi=1000)
```

```
<matplotlib.figure.Figure at 0x11bad3cd0>
```



- Think about the implications of data analysis and the relation to networks. Try to come with a domain of interest to apply tools of network science.

- Search in the Internet network science applicatior for that topic.

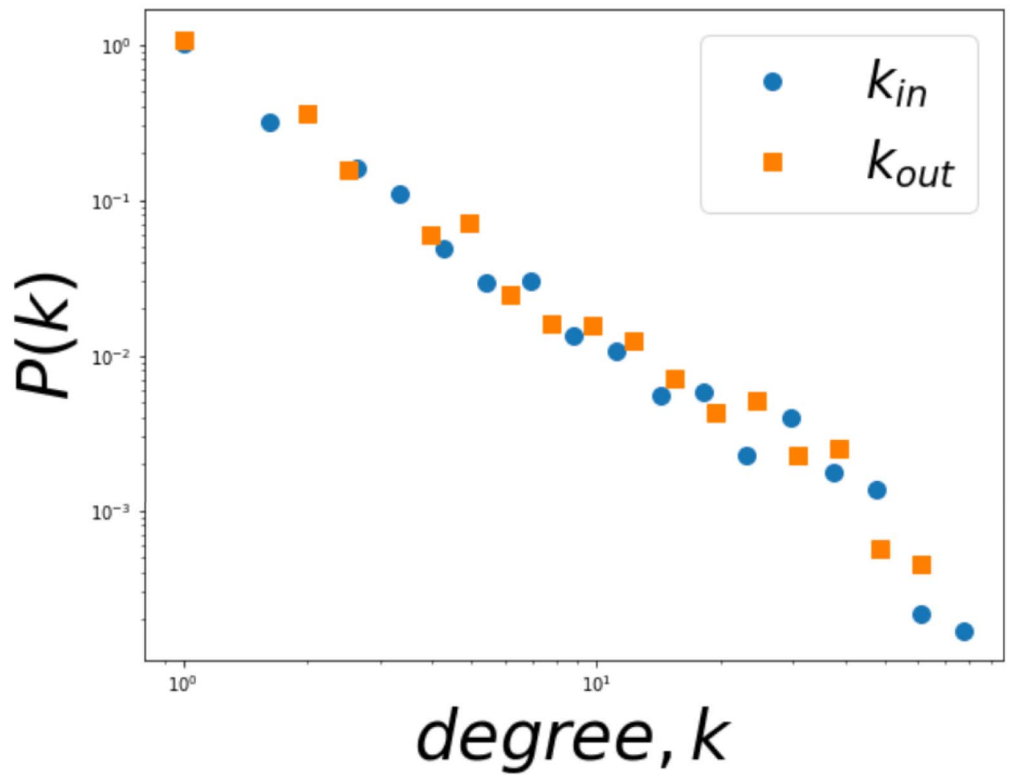
In [73]:
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
try:
    import powerlaw
    import mpmath
except:
    !pip install powerlaw
    !pip install mpmath
    import mpmath
    import powerlaw
import scipy as sp
%matplotlib inline
```

In [74]:
```
def makeNetworkFromFiles(fedges, fnodes,deli):
    ''' Generate a network from csv files.
    The first column of the nodes file is taken to be the id by default.
    The nodes file must contain columns named weight, source, and target.
    The deli parameter is the delimeter of the file. '''
    G = nx.DiGraph()  #modified from lecture 12
    fn=pd.read_csv(fnodes,delimiter=deli,index_col=0).transpose()
    n_attr=len(fn.index.values)
    attr=fn.index.values
    for n in fn.columns:
        attr_node=dict(zip(attr, fn[n].values))
        G.add_node(n,attr_dic=attr_node)
    fl=pd.read_csv(fedges,delimiter=deli)
    fl.columns = map(str.lower, fl.columns)
    for L in fl.index.values:
        G.add_edge(fl['source'][L],fl['target'][L],weight=fl['weight'][L])
    #GL = max(nx.connected_component_subgraphs(G), key=len) ## no implemented for directed graph
    return G
```

In [53]:
```
def remove_values_from_list(the_list, val):
    return [value for value in the_list if value != val]
```

```
#data = np.array(pd.read_csv('THE_LINKS.txt',sep=' ',header=0))
#g = nx.DiGraph()
#for row in data:
 #   g.add_edge(row[0],row[1],weight=row[3])
degrees = []
out_weighted = []
out_d= []
in_weighted = []
in_d= []
g= makeNetworkFromFiles('Air_Data [Edges].csv', 'Air_Data [Nodes].csv',',')
#g= makeNetworkFromFiles('SchoolEdges.csv', 'SchoolNodes.csv',' ')
#g=makeNetworkFromFiles('USCommuting_LINKS.csv', 'USCommuting_NODES.csv',' ')

degrees=dict(g.degree()).values()
out_d=dict(g.out_degree()).values()
in_d=dict(g.in_degree()).values()

######NEED to eliminate ZEROS
in_d=remove_values_from_list(in_d,0)
out_d=remove_values_from_list(out_d,0)

out_weighted=dict(g.out_degree(weight='weight')).values()
in_weighted=dict(g.in_degree(weight='weight')).values()
######NEED to elkminate ZEROS
out_weighted=remove_values_from_list(out_weighted,0)
in_weighted=remove_values_from_list(in_weighted,0)
```

```
fig, ax = plt.subplots()
fig.set_size_inches((9, 7))

n_bins = 20

#n, bins = np.histogram(out_d, bins = range(min(out_d), max(out_d)+1, 2), normed="True")
out_logBins = np.logspace(np.log10(min(out_d)), np.log10(max(out_d)),num=n_bins)
out_logBinDensity, out_binedges = np.histogram(out_d, bins=out_logBins, density=True)

#n, bins = np.histogram(in_d, bins = range(min(in_d), max(in_d)+1, 2), normed="True")
in_logBins = np.logspace(np.log10(min(in_d)), np.log10(max(in_d)),num=n_bins)
in_logBinDensity, in_binedges = np.histogram(in_d, bins=in_logBins, density=True)

ax.loglog(out_logBins[:-1],out_logBinDensity,'o', markersize=10,label=r'$k_{in}$')
ax.loglog(in_logBins[:-1],in_logBinDensity,'s', markersize=10,label=r'$k_{out}$')
ax.legend(fontsize=30)


ax.set_xlabel('$degree, k$',fontsize=40)
ax.set_ylabel('$P(k)$',fontsize=40)
plt.savefig("distributions.eps",dpi=200,bbox_inches='tight')
```



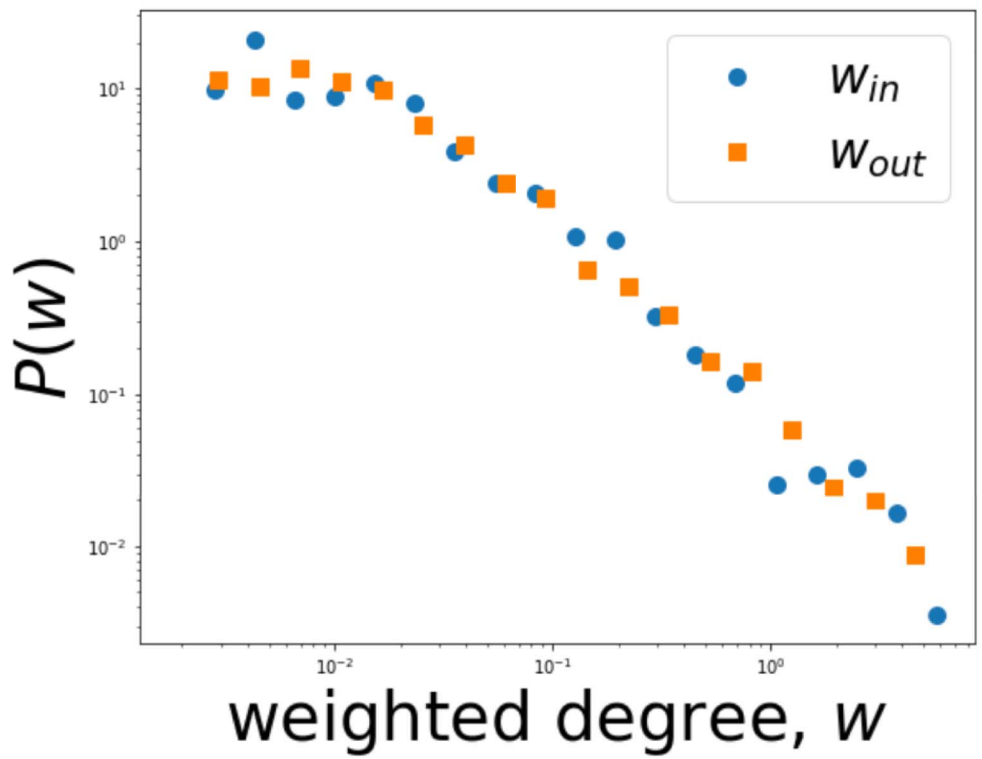

```
len(g.nodes())
```

```
332
```

```
len(g.edges())
```

```
2126
```

```
fig, ax = plt.subplots()
fig.set_size_inches((9, 7))

n_bins = 20

#n, bins = np.histogram(out_weighted, bins = range(min(out_weighted), max(out_weighted)+1, 2), normed="True")
out_logBins = np.logspace(np.log10(min(out_weighted)+0.001), np.log10(max(out_weighted)),num=n_bins)
out_logBinDensity, out_binedges = np.histogram(out_weighted, bins=out_logBins, density=True)

#n, bins = np.histogram(in_weighted, bins = range(min(in_weighted), max(in_weighted)+1, 2), normed="True")
in_logBins = np.logspace(np.log10(min(in_weighted)+0.001), np.log10(max(in_weighted)),num=n_bins)
in_logBinDensity, in_binedges = np.histogram(in_weighted, bins=in_logBins, density=True)

ax.loglog(out_logBins[:-1],out_logBinDensity,'o', markersize=10,label=r'$w_{in}$')
ax.loglog(in_logBins[:-1],in_logBinDensity,'s', markersize=10,label=r'$w_{out}$')
ax.legend(fontsize=30)


ax.set_xlabel('weighted degree, $w$',fontsize=40)
ax.set_ylabel('$P(w)$',fontsize=40)
plt.savefig("weighted.eps",dpi=200,bbox_inches='tight')
```



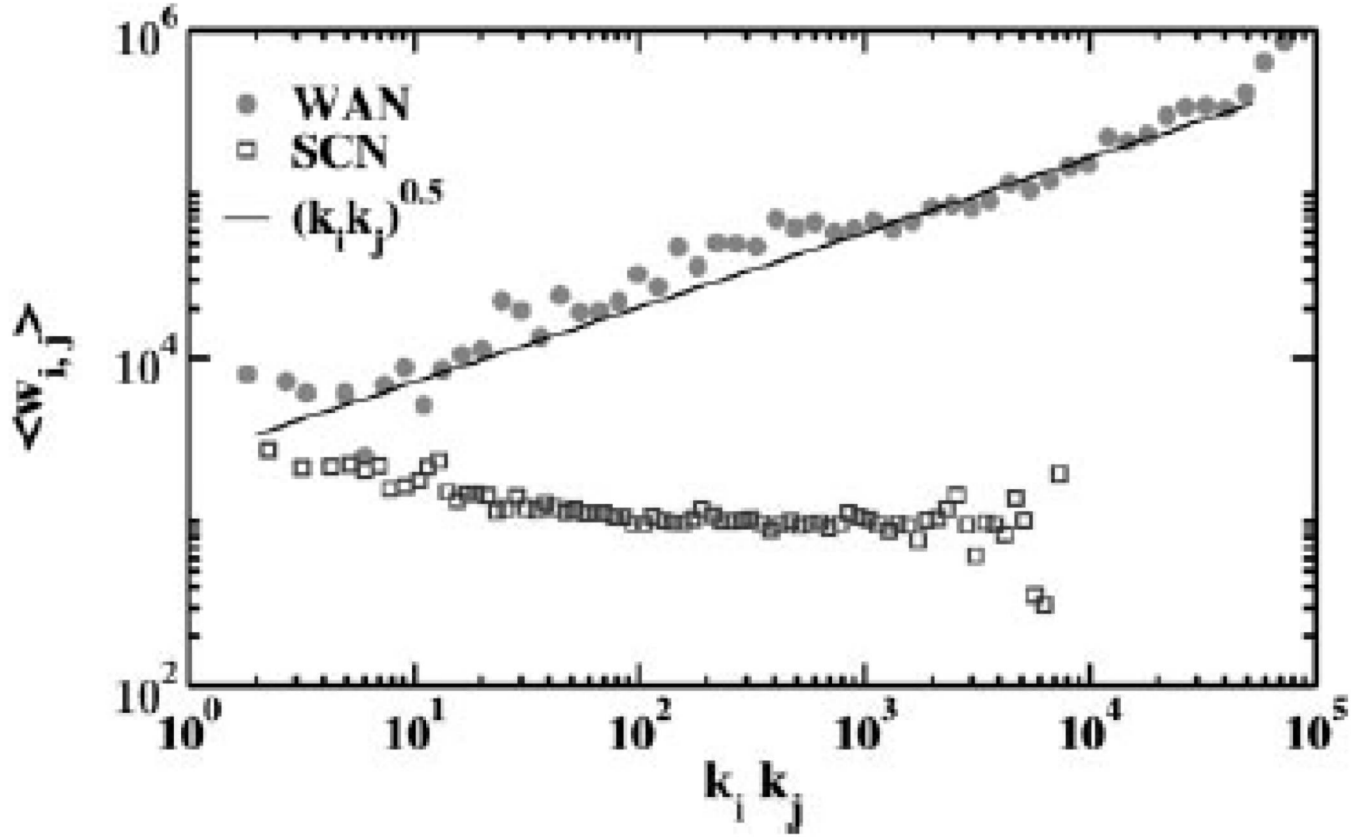


Fig. 4. Average weight as a function of the end-point degree. The solid line corresponds to a power-law behavior $\langle w_{ij} \rangle \sim (k_i k_j)^{\theta}$, with exponent $\theta = 0.5 \pm 0.1$. In the case of the SCN it is possible to observe an almost flat behavior for roughly two orders of magnitude.

```
kk = []
wij = []
degrees = g.degree()
for n in g.nodes(data=True):
    for e in g.edges(n[0],data=True):
        kk.append(degrees[e[0]]*degrees[e[1]])
        wij.append(g[e[0]][e[1]]['weight'])
```

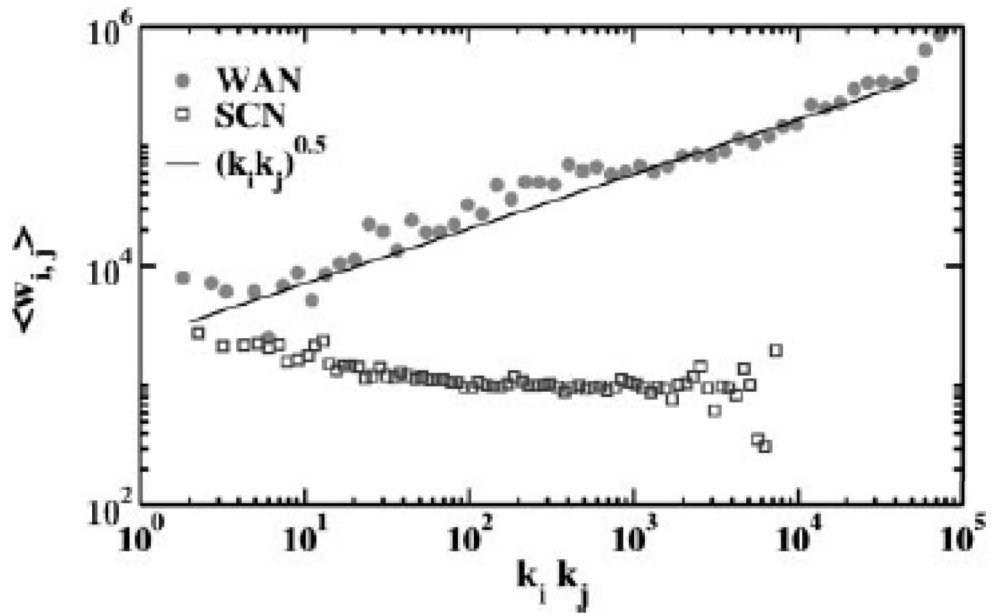


Fig. 4. Average weight as a function of the end-point degree. The solid line corresponds to a power-law behavior $\langle w_{ij} \rangle \sim (k_i k_j)^{\theta}$, with exponent $\theta = 0.5 \pm 0.1$. In the case of the SCN it is possible to observe an almost flat behavior for roughly two orders of magnitude.

[illegible]

numpy.histogram

`numpy.` **histogram**(*a*, *bins=10*, *range=None*, *normed=False*, *weights=None*, *density=None*) [source]

Compute the histogram of a set of data.

Parameters:

a *: array_like*

Input data. The histogram is computed over the flattened array.

bins *: int or sequence of scalars or str, optional*

If *bins* is an int, it defines the number of equal-width bins in the given range (10, by default). If *bins* is a sequence, it defines the bin edges, including the rightmost edge, allowing for non-uniform bin widths.

New in version 1.11.0.

If *bins* is a string from the list below, `histogram` will use the method chosen to calculate the optimal bin width and consequently the number of bins (see *Notes* for more detail on the estimators) from the data that falls within the requested range. While the bin width will be optimal for the actual data in the range, the number of bins will be computed to fill the entire range, including the empty portions. For visualisation, using the 'auto' option is suggested. Weighted data is not supported for automated bin size selection.

'auto'

Maximum of the 'sturges' and 'fd' estimators. Provides good all around performance.

'fd' (Freedman Diaconis Estimator)

Robust (resilient to outliers) estimator that takes into account data variability and data size.

'doane'

An improved version of Sturges' estimator that works better with non-normal datasets.

'scott'

Less robust estimator that that takes into account data variability and data size.

'rice'

Estimator does not take variability into account, only data size. Commonly overestimates number of bins required.

'sturges'

R's default method, only accounts for data size. Only optimal for gaussian data and underestimates number of bins for large non-gaussian datasets.

'sqrt'

Square root (of data size) estimator, used by Excel and other programs for its speed and simplicity.

range *: (float, float), optional*

```
n_bins = 40
kk_logBins = np.logspace(np.log10(min(kk)), np.log10(max(kk)),num=n_bins)
counts, bins = np.histogram(kk, bins=kk_logBins);
sums, bins = np.histogram(kk, bins=kk_logBins,weights=wij);
avg_w = sums/counts;
```

```
fig, ax = plt.subplots()
fig.set_size_inches((18, 7))
ax.loglog(bins[:-1],avg_w,linewidth=0,color='r',marker='o',markersize=10)
ax.set_xlabel('$k_ik_j$',fontsize=20)
ax.set_ylabel('$<w_{i,j}>$',fontsize=20)
plt.savefig("commutes_directed_.eps",dpi=200,bbox_inches='tight')
```