DATA

```
!wget http://stat-computing.org/dataexpo/2009/carriers.csv
!wget http://stat-computing.org/dataexpo/2009/plane-data.csv
!wget http://stat-computing.org/dataexpo/2009/airports.csv
```

```
--2021-10-24 14:26:57--  http://stat-computing.org/dataexpo/2009/carriers.csv
Resolving stat-computing.org (stat-computing.org)... 52.218.128.139
Connecting to stat-computing.org (stat-computing.org)|52.218.128.139|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 43758 (43K) [text/csv]
Saving to: ‘carriers.csv’

carriers.csv        100%[===================>]  42.73K  --.-KB/s    in 0.08s

2021-10-24 14:26:57 (533 KB/s) - ‘carriers.csv’ saved [43758/43758]

--2021-10-24 14:26:57--  http://stat-computing.org/dataexpo/2009/plane-data.csv
Resolving stat-computing.org (stat-computing.org)... 52.218.128.139
Connecting to stat-computing.org (stat-computing.org)|52.218.128.139|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 428796 (419K) [text/csv]
Saving to: ‘plane-data.csv’

plane-data.csv      100%[===================>] 418.75K  2.37MB/s    in 0.2s

2021-10-24 14:26:58 (2.37 MB/s) - ‘plane-data.csv’ saved [428796/428796]

--2021-10-24 14:26:58--  http://stat-computing.org/dataexpo/2009/airports.csv
Resolving stat-computing.org (stat-computing.org)... 52.218.128.139
Connecting to stat-computing.org (stat-computing.org)|52.218.128.139|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 244438 (239K) [text/csv]
Saving to: ‘airports.csv’

airports.csv        100%[===================>] 238.71K  1.48MB/s    in 0.2s

2021-10-24 14:26:58 (1.48 MB/s) - ‘airports.csv’ saved [244438/244438]
```

```
!wget https://raw.githubusercontent.com/KeplerC/Mathematical-Networks/master/Airport-Delay-Analysis/Data/airport_codes.csv
```

```
--2021-10-24 14:26:58--  https://raw.githubusercontent.com/KeplerC/Mathematical-Networks/master/Airport-Delay-Analysis/Data/airport_codes.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 33054 (32K) [text/plain]
Saving to: ‘airport_codes.csv’

airport_codes.csv   100%[===================>]  32.28K  --.-KB/s    in 0.001s

2021-10-24 14:26:59 (37.5 MB/s) - ‘airport_codes.csv’ saved [33054/33054]
```

```
!wget http://stat-computing.org/dataexpo/2009/2008.csv.bz2
```

```
--2019-06-01 03:48:33--  http://stat-computing.org/dataexpo/2009/2008.csv.bz2
Resolving stat-computing.org (stat-computing.org)... 52.218.193.11
Connecting to stat-computing.org (stat-computing.org)|52.218.193.11|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 113753229 (108M) [application/x-bzip2]
Saving to: ‘2008.csv.bz2’

2008.csv.bz2        100%[===================>] 108.48M  32.4MB/s    in 3.5s

2019-06-01 03:48:37 (30.9 MB/s) - ‘2008.csv.bz2’ saved [113753229/113753229]
```

```
!mkdir Data
!mv *.csv ./Data
```

```
mkdir: cannot create directory ‘Data’: File exists
```

```
!bzip2 -d 2008.csv.bz2
```

```
!mkdir ./Data/trip
!mv ./2008.csv ./Data/trip
```

```
mkdir: cannot create directory ‘./Data/trip’: File exists
```

```
from __future__ import division
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
import seaborn as sns
import networkx as nx
import numpy as np
import os
# http://stat-computing.org/dataexpo/2009/the-data.html
```

Loading...

About the following imports:

- *airports* contains coordinates for airports in the USA.
- *carriers* contains airport names and cities that are useful for data merging and indexing.
- *codes* contains airport codes useful for merging and indexing (Source:http://www.airportcodes.us/us-airports.htm)

```
# Data imports (files attached in the email assignment)
carriers  = pd.read_csv('Data/carriers.csv')
carriers.columns = ['UniqueCarrier','Description']
carriers.dropna(how="all", inplace=True)
carriers.head()
```

	UniqueCarrier	Description
0	02Q	Titan Airways
1	04Q	Tradewind Aviation
2	05Q	Comlux Aviation, AG
3	06Q	Master Top Linhas Aereas Ltd.
4	07Q	Flair Airlines Ltd.

```
codes = pd.read_csv('Data/airport_codes.csv')
codes.head()
```

	Code	Name	City	State
0	0AK	Pilot Station Airport	Pilot Station	AK
1	16A	Nunapitchuk Airport	Nunapitchuk	AK
2	1G4	Grand Canyon West Airport	Peach Springs	AZ
3	2A3	Larsen Bay Airport	Larsen Bay	AK
4	2A9	Kotlik Airport	Kotlik	AK

```
codes = codes.set_index('Code').to_dict()
```

```
airports  = pd.read_csv('Data/airports.csv')
airports = airports.query("country == 'USA'")
airports.head()
```

	iata	airport	city	state	country	lat	long
0	00M	Thigpen	Bay Springs	MS	USA	31.953765	-89.234505
1	00R	Livingston Municipal	Livingston	TX	USA	30.685861	-95.017928
2	00V	Meadow Lake	Colorado Springs	CO	USA	38.945749	-104.569893
3	01G	Perry-Warsaw	Perry	NY	USA	42.741347	-78.052081
4	01J	Hilliard Airpark	Hilliard	FL	USA	30.688012	-81.905944

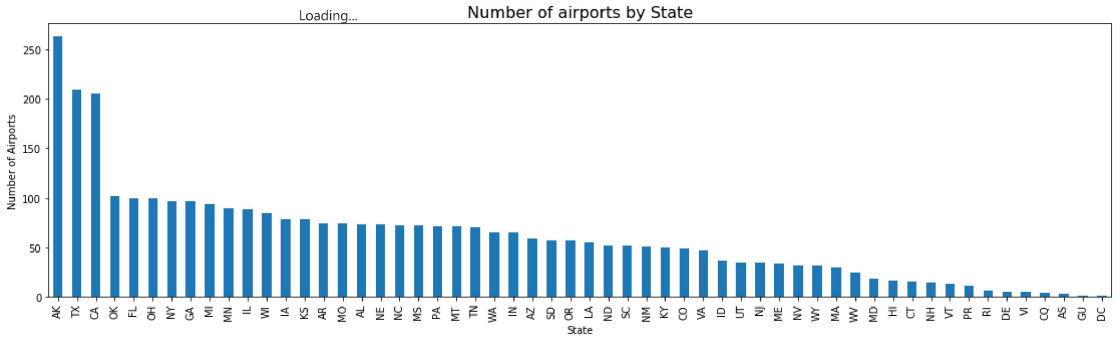
Basic exploration

Just out of curiosity, I plotted the number of airports in each State. There are big states with not that many airports, and smaller states with a large amount of airports.

```
plt.figure(figsize=(19,5))
```

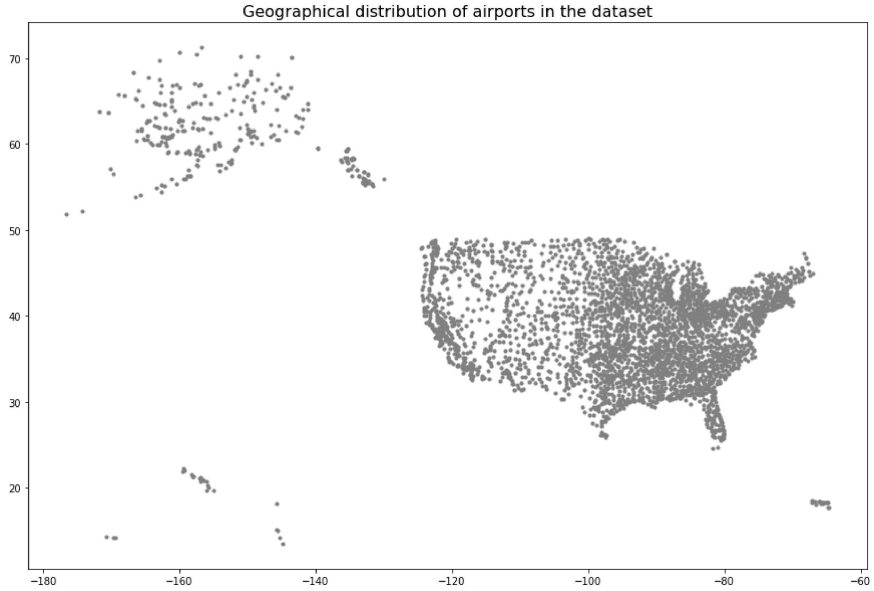
```
airports.state.value_counts().plot(kind='bar')
plt.title('Number of airports by State',fontsize=16)
plt.xlabel('State')
plt.ylabel('Number of Airports')
```

```
Text(0,0.5,'Number of Airports')
```



Plotting the distribution of airportswhich follows the known patterns in US American geography and political divisions.

```
plt.figure(figsize=(15,10))
plt.scatter(airports.long,airports.lat,s=10,c='grey')
plt.title('Geographical distribution of airports in the dataset',fontsize=16)
plt.show()
```



For the centrality index, only be working with 2008 data.

```
trips_08 = pd.read_csv('Data/trip/2008.csv',usecols=['Origin','Dest','UniqueCarrier','Year','ActualElapsedTime','CRSElapsedTime','ArrDelay','DepDelay','Distance','Cancelled'])
trips_08.head()
```

	Year	UniqueCarrier	ActualElapsedTime	CRSElapsedTime	ArrDelay	DepDelay	Origin	Dest	Distance	Cancelled
0	2008	WN	128.0	150.0	-14.0	8.0	IAD	TPA	810	0

```
print("The total number of flights was " + str(len(trips_08)) + " whereas the number of not cancelled trips was " +str(trips_08.Cancelled.value_counts()[0]) + ". The percentage of cancellation is " + str(100 - 100*trips_08.Cancelled.value_counts()[0] / len(trips_08)) +
```

```
The total number of flights was 7009728 whereas the number of not cancelled trips was 6872294. The percentage of cancellation is 1.9606181580797397 %
```

Loading...

In the following lines, created a single data set with carriers, locations and trips:

```
airports = airports[['airport','city','lat','long']].set_index('city').to_dict() # Useful for mapping airports to coordinates

data = trips_08.merge(carriers,on='UniqueCarrier')
data["origin_city"] = data["Origin"].map(codes['City'])
data["destin_city"] = data["Dest"].map(codes['City'])
data["origin_lat"]  = data["origin_city"].map(airports['lat'])
data["origin_long"] = data["origin_city"].map(airports['long'])
data["destin_lat"]  = data["destin_city"].map(airports['lat'])
data["destin_long"] = data["destin_city"].map(airports['long'])
```

The following data set comprises the average arrival, departure and total delay between every existing origin-destination pair in 2008:

```
# Average delays between airports
delays = data.groupby(['origin_city','destin_city'])[['ArrDelay','DepDelay']].mean().reset_index()
delays['Total_delay'] = delays.ArrDelay+delays.DepDelay
delays.head()
```

	origin_city	destin_city	ArrDelay	DepDelay	Total_delay
0	Abilene	Dallas	5.037490	6.880397	11.917888
1	Adak Island	Anchorage	4.532609	13.652174	18.184783
2	Aguadilla	New York	-0.175038	0.811263	0.636225
3	Aguadilla	Newark	6.271978	11.863014	18.134992
4	Aguadilla	Orlando	2.563786	6.557377	9.121163

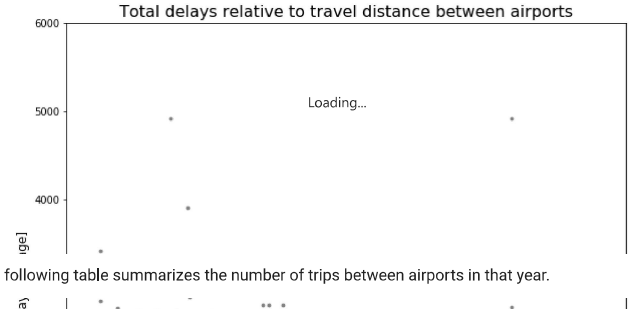
```
print('Airport with lenghtiest delays at departure: ' + delays.groupby('origin_city')['DepDelay'].mean().idxmax())
print('Airport with lenghtiest delays at arrival: ' + delays.groupby('destin_city')['ArrDelay'].mean().idxmax())
print('Airport with lenghtiest total delays: ' + delays.groupby('destin_city')['Total_delay'].mean().idxmax())
```

```
Airport with lenghtiest delays at departure: Champaign
Airport with lenghtiest delays at arrival: Springfield
Airport with lenghtiest total delays: Springfield
```

More notes on exploration: this is a simple (and naive) plot comparing distance between airports and total delays.

```
plt.figure(figsize=(10,10))
plt.scatter(data.Distance,data.ArrDelay + data.DepDelay, c = 'grey', s = 8)
plt.xlim([0,5000])
plt.ylim([-200,6000])
plt.title('Total delays relative to travel distance between airports',fontsize=16)
plt.xlabel('Distance [miles]',fontsize=12)
plt.ylabel('Total delay [average]',fontsize=12)
```

```
Text(0,0.5,'Total delay [average]')
```



The following table summarizes the number of trips between airports in that year.

```
trips = data.groupby(['origin_city','destin_city'])
trips = trips[['UniqueCarrier']].count().reset_index()
trips.head()
```

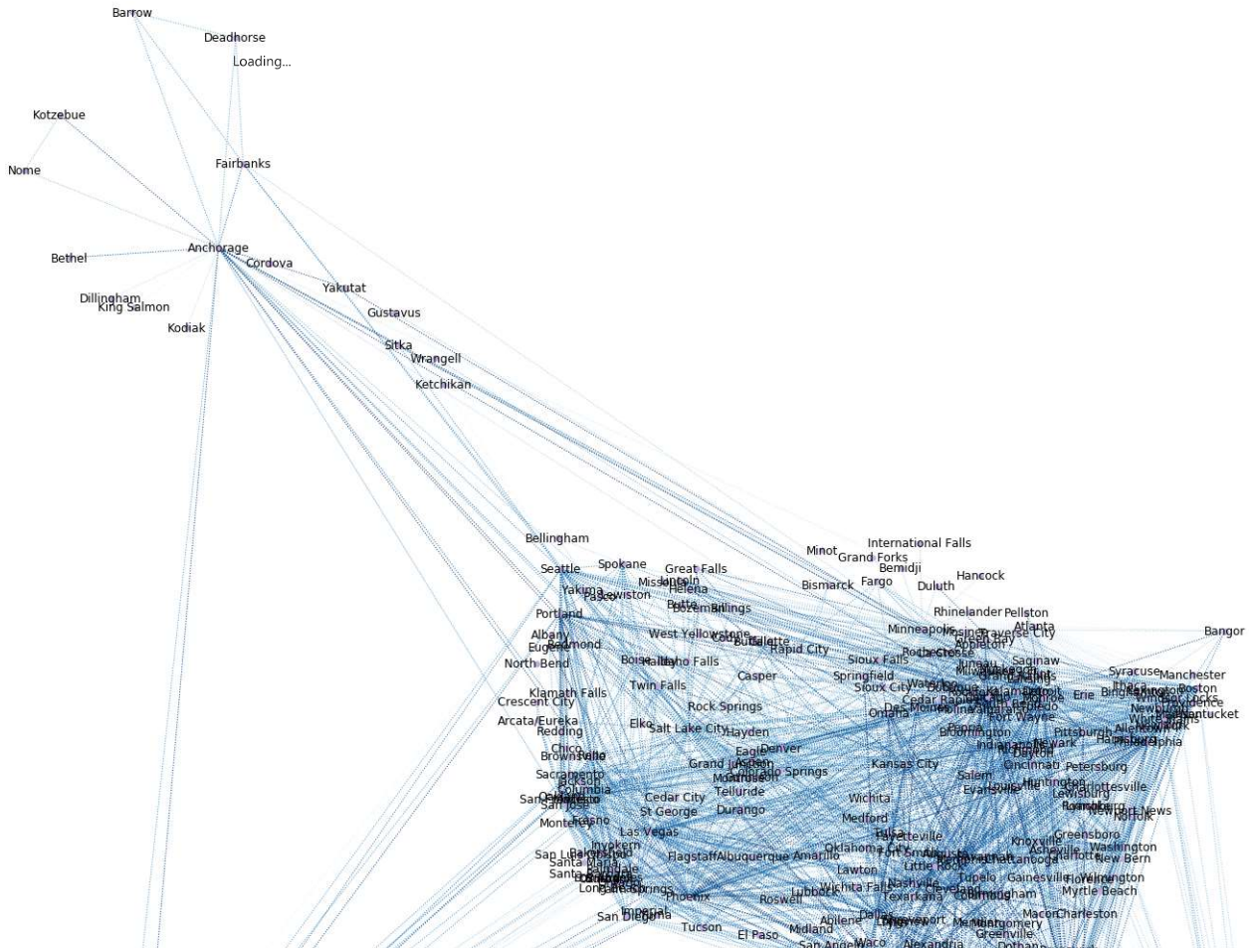
	origin_city	destin_city	UniqueCarrier
0	Abilene	Dallas	2660
1	Adak Island	Anchorage	102
2	Aguadilla	New York	661
3	Aguadilla	Newark	366
4	Aguadilla	Orlando	492

Representing this data set as a network, where edges are trips between airports, weights are the number of trips and the nodes are airports.

```
# Some data cleaning and preparation
data = data[data['origin_lat'].notnull()]
data = data[data['origin_long'].notnull()]
data['coord'] = zip(data['origin_long'],data['origin_lat'])


# Location of the nodes
pos  = data[['origin_city','coord']].dropna().set_index('origin_city').to_dict()
# Network representation
G = nx.Graph()
G.add_nodes_from(list(pd.unique(trips.origin_city.dropna())))
G.add_weighted_edges_from(zip(trips['origin_city'].dropna(), trips['destin_city'].dropna(), trips['UniqueCarrier'].dropna()))
```

The following is a (not very appealing) graph of all the trips between airports.

```
## G.remove_nodes_from(['Keahole','Orange County','Gulfport','North Canton','Elmira / Corning','Champaign','Fort Lauderdale','Rdu Airport','Avoca','Saint Petersburg','Saint Louis','Blountville','BWI Airport','Fort Myers','Mcallen','Pocatello','South Burlington'])
G.remove_nodes_from(['Fort Hood / Killeen','Pierre','Pueblo','Ogden','Keahole','Fayetteville / Springdale','Adak Island','Egg Harbor Township','Cheyenne','Beaumont / Port Arthur','Champaign','Orange County','BWI Airport','Gulfport','South Burlington','North Canton','El
plt.figure(figsize = (20,20))

nx.draw(G,pos = pos['coord'],with_labels=True,node_size=30,style='dotted', edge_color=range(G.number_of_edges()), edge_cmap=plt.cm.Blues,node_color='#C3ABE2')
```




```
d = dict(G.degree()).values()
j = np.argmax(d)
print('The most connected airport is {0} with {1} connections'.format(dict(G.degree()).keys()[j],d[j]))
```

```
The most connected airport is Atlanta with 159 connections
```

```
from scipy.stats import *
```

Representing the airports in terms of their betweeness centrality relative to their average arrival and departure delays.

Top airports in terms of their betweenness centrality for domestic flights (similar for arrivals and departures):

```
delays = delays[delays.destin_city != "Pierre"]
ar_delays  = delays.groupby(['destin_city'])['ArrDelay'].mean().reset_index()
dep_delays = delays.groupby(['origin_city'])['DepDelay'].mean().reset_index()

# one could argue it is the other way around
sizes_ar = nx.betweenness_centrality(G)
factor=1.0/sum(sizes_ar.itervalues())
for k in sizes_ar:
    sizes_ar[k] = sizes_ar[k]*factor
sizes_dep = sizes_ar
temp = pd.DataFrame([]).from_dict(sizes_ar, orient='index')
```

```
temp.columns = ['Betweenness Centrality']
temp.sort_values('Betweenness Centrality', ascending=False).head(10)
```

	Betweenness Centrality
Atlanta	0.132375
Salt Lake City	0.085970
Dallas	0.083395
Minneapolis	0.078855
Anchorage	0.058678
Chicago	0.057493
Denver	0.052610
Houston	0.045226
Detroit	0.044893
San Francisco	0.035316

Loading...

```
import random
```

```
import random
temp['Arrival Delay'] = 10
temp = temp[temp["Betweenness Centrality"] != 0]
for k in temp.index:
  for index, row in ar_delays.iterrows():
    if row["destin_city"] == k:
      try:
        temp["Arrival Delay"][k] = row["ArrDelay"]
        temp["Betweenness Centrality"][k] += random.random() *  row["ArrDelay"]  / 1000

      except:
        print(k)

temp['Departure Delay'] = 10
temp = temp[temp["Betweenness Centrality"] != 0]
for k in temp.index:
  for index, row in dep_delays.iterrows():
    if row["origin_city"] == k:
      try:
        temp["Departure Delay"][k] = row["DepDelay"]
        temp["Betweenness Centrality"][k] += random.random() *  row["DepDelay"]  / 1000

      except:
        print(k)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

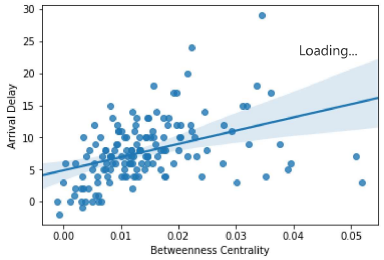
```
temp = temp[temp["Betweenness Centrality"] < 0.06] [temp["Arrival Delay"] < 50]
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  """Entry point for launching an IPython kernel.
```

```
import seaborn as sns
```

```
sns.regplot(x = "Betweenness Centrality", y = "Arrival Delay", data = temp)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc073e34c90>
```

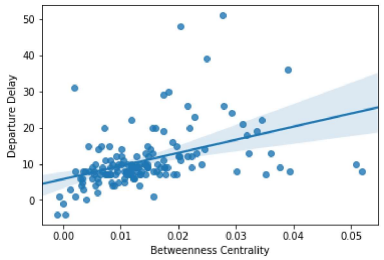



```
pearsonr(temp["Betweenness Centrality"], temp["Arrival Delay"]), spearmanr(temp["Betweenness Centrality"], temp["Arrival Delay"]),
```

```
((0.41367885290030043, 1.732373741382247e-07),
 SpearmanrResult(correlation=0.4888453369146027, pvalue=2.905983137719665e-10))
```

```
import seaborn as sns

sns.regplot(x = "Betweenness Centrality", y = "Departure Delay", data = temp)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc073e27d10>
```




```
pearsonr(temp["Betweenness Centrality"], temp["Departure Delay"]), spearmanr(temp["Betweenness Centrality"], temp["Departure Delay"]),
```

```
((0.4302702630280377, 4.814440047359346e-08),
 SpearmanrResult(correlation=0.5355810107086307, pvalue=2.320150843653816e-12))
```

```
delays = delays[delays.destin_city != "Pierre"]
ar_delays  = delays.groupby(['destin_city'])['ArrDelay'].mean().reset_index()
dep_delays = delays.groupby(['origin_city'])['DepDelay'].mean().reset_index()

# one could argue it is the other way around
sizes_ar = nx.closeness_centrality(G)
factor=1.0/sum(sizes_ar.itervalues())
for k in sizes_ar:
    sizes_ar[k] = sizes_ar[k]*factor
sizes_dep = sizes_ar
temp = pd.DataFrame([]).from_dict(sizes_ar, orient='index')
temp.columns = ['Closeness Centrality']
temp.sort_values('Closeness Centrality', ascending=False).head(10)
```

	Closeness Centrality
Atlanta	0.005457
Chicago	0.005242
Dallas	0.005112
Denver	0.005010

Loading...

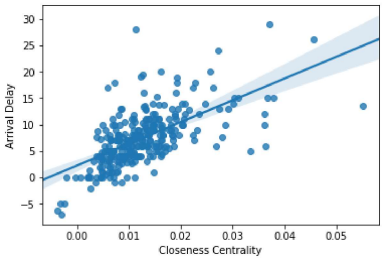
```
import random
```

```
import random
temp['Arrival Delay'] = 10
temp = temp[temp["Closeness Centrality"] != 0]
for k in temp.index:
  for index, row in ar_delays.iterrows():
    if row["destin_city"] == k:
      try:
        temp["Arrival Delay"][k] = row["ArrDelay"]
        temp["Closeness Centrality"][k] += random.random() *  row["ArrDelay"]  / 1000

      except:
        print(k)

temp['Departure Delay'] = 10
temp = temp[temp["Closeness Centrality"] != 0]
for k in temp.index:
  for index, row in dep_delays.iterrows():
    if row["origin_city"] == k:
      try:
        temp["Departure Delay"][k] = row["DepDelay"]
        temp["Closeness Centrality"][k] += random.random() *  row["DepDelay"]  / 1000

      except:
        print(k)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
import seaborn as sns
temp = temp[temp["Arrival Delay"] < 50]
sns.regplot(x = "Closeness Centrality", y = "Arrival Delay", data = temp)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc0740d04d0>
```



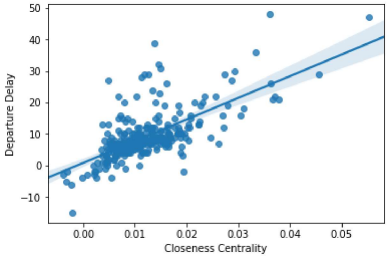

```
pearsonr(temp["Closeness Centrality"], temp["Arrival Delay"]), spearmanr(temp["Closeness Centrality"], temp["Arrival Delay"]),
```

```
((0.6342169562881474, 5.019220316642398e-33),
 SpearmanrResult(correlation=0.6443133515447522, pvalue=2.351121848583335e-34))
```

```
import seaborn as sns
temp = temp[temp["Departure Delay"] < 50]
sns.regplot(x = "Closeness Centrality", y = "Departure Delay", data = temp)
```

Loading...

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc073de7d90>
```




```
pearsonr(temp["Closeness Centrality"], temp["Departure Delay"]), spearmanr(temp["Closeness Centrality"], temp["Departure Delay"]),
```

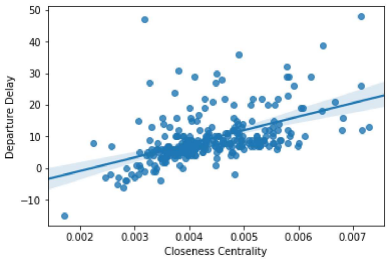
```
((0.6755507560803957, 8.360383252503919e-39),
 SpearmanrResult(correlation=0.6337421655044162, pvalue=5.7801571660847385e-33))
```

```
import random
temp['Departure Delay'] = 10
temp = temp[temp["Closeness Centrality"] != 0]
for k in temp.index:
  for index, row in dep_delays.iterrows():
    if row["origin_city"] == k:
      try:
        temp["Departure Delay"][k] = row["DepDelay"]
        temp["Closeness Centrality"][k] += random.random() *  row["DepDelay"]  / 10000

      except:
        print(k)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

```
import seaborn as sns
temp = temp[temp["Departure Delay"] < 50]
sns.regplot(x = "Closeness Centrality", y = "Departure Delay", data = temp)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc074238d10>
```



```
delays = delays[delays.destin_city != "Pierre"]
ar_delays  = delays.groupby(['destin_city'])['ArrDelay'].mean().reset_index()
dep_delays = delays.groupby(['origin_city'])['DepDelay'].mean().reset_index()

# one could argue it is the other way around
sizes_ar = nx.degree_centrality(G)
factor=1.0/sum(sizes_ar.itervalues())
for k in sizes_ar:
    sizes_ar[k] = sizes_ar[k]*factor
sizes_dep = sizes_ar
temp = pd.DataFrame([]).from_dict(sizes_ar, orient='index')
temp.columns = ['Degree Centrality']
temp.sort_values('Degree Centrality', ascending=False).head(10)
```

Loading...

	Degree Centrality
Atlanta	0.030448
Chicago	0.027193
Dallas	0.025086
Denver	0.023363
Minneapolis	0.022980
Detroit	0.021448
Salt Lake City	0.021256
Houston	0.021256
Cincinnati	0.020490
New York	0.018384

```
import random
```

```
import random
temp['Arrival Delay'] = 10
temp = temp[temp["Degree Centrality"] != 0]
for k in temp.index:
  for index, row in ar_delays.iterrows():
    if row["destin_city"] == k:
      try:
        temp["Arrival Delay"][k] = row["ArrDelay"]
        temp["Degree Centrality"][k] += random.random() *  row["ArrDelay"]  / 10000

      except:
        print(k)

temp['Departure Delay'] = 10
temp = temp[temp["Degree Centrality"] != 0]
for k in temp.index:
  for index, row in dep_delays.iterrows():
    if row["origin_city"] == k:
      try:
        temp["Departure Delay"][k] = row["DepDelay"]
        temp["Degree Centrality"][k] += random.random() *  row["DepDelay"]  / 10000

      except:
        print(k)
```

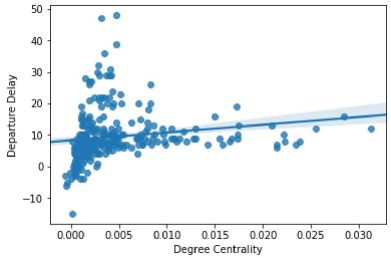
```
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

```
import seaborn as sns
temp = temp[temp["Departure Delay"] < 50]
sns.regplot(x = "Degree Centrality", y = "Departure Delay", data = temp)
```

Loading...

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc074e0ee10>
```

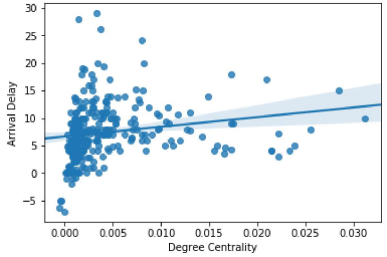



```
pearsonr(temp["Degree Centrality"], temp["Departure Delay"]), spearmanr(temp["Degree Centrality"], temp["Departure Delay"]),
```

```
SpearmanrResult(correlation=0.5183102692455649, pvalue=1.0136604108893987e-20)
```

```
import seaborn as sns
temp = temp[temp["Arrival Delay"] < 50]
sns.regplot(x = "Degree Centrality", y = "Arrival Delay", data = temp)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc0740e7e10>
```




```
pearsonr(temp["Degree Centrality"], temp["Arrival Delay"]), spearmanr(temp["Degree Centrality"], temp["Arrival Delay"]),
```

```
((0.1787280836167976, 0.002639077479666188),
 SpearmanrResult(correlation=0.40416213007312046, pvalue=1.8183775677844777e-12))
```

```
delays = delays[delays.destin_city != "Pierre"]
ar_delays  = delays.groupby(['destin_city'])['ArrDelay'].mean().reset_index()
dep_delays = delays.groupby(['origin_city'])['DepDelay'].mean().reset_index()

# one could argue it is the other way around
sizes_ar = nx.eigenvector_centrality(G)
factor=1.0/sum(sizes_ar.itervalues())
for k in sizes_ar:
    sizes_ar[k] = sizes_ar[k]*factor
sizes_dep = sizes_ar
temp = pd.DataFrame([]).from_dict(sizes_ar, orient='index')
temp.columns = ['Katz Centrality']
temp.sort_values('Katz Centrality', ascending=False).head(10)
```

	Katz Centrality
Atlanta	0.018876
Chicago	0.018808
Dallas	0.017695
Denver	0.016908
Houston	0.016778

Loading...

Worst performing airports in terms of arrival delays:

```
ar_delays.sort_values('ArrDelay',ascending=False).head(10)
```

	destin_city	ArrDelay
254	Springfield	63.447158
12	Aspen	29.995056
106	Gunnison	28.909466
183	Nantucket	26.079958
68	Des Moines	24.121120
193	North Bend	20.736623
192	Norfolk	20.653845
51	Chico	19.905077
75	Eagle	19.462659
39	Butte	18.991971

Worst performing airports in terms of average departure delays:

```
dep_delays.sort_values('DepDelay',ascending=False).head(10)
```

	origin_city	DepDelay
43	Champaign	149.266698
225	Roanoke	51.956570
10	Arcata/Eureka	48.000720
135	Klamath Falls	47.066221
276	Wichita	39.904703
9	Appleton	36.742313
47	Charlottesville	32.597403
98	Grand Junction	31.171125
83	Fargo	30.783061
182	Nantucket	29.892045

	Katz Centrality
Roanoke	0.002297
Albany	0.005028
St George	0.000770
Montgomery	0.001563
Newark	0.015544
Hayden	0.003361
Tyler	0.000385
Hailey	0.000345
Hilo	0.000203
Gunnison	0.001890
Minot	0.000362
Bloomington	0.002034
Brunswick	0.000411
Lynchburg	0.000411
Yuma	0.001364
West Yellowstone	0.000316
Columbia	0.003593
Ketchikan	0.000258
King Salmon	0.000117
Harrisburg	0.003894
Lihue	0.001957
Sacramento	0.008062
Charleston	0.006003
El Paso	0.004508
Santa Maria	0.000720
Fort Wayne	0.001974
Flint	0.003425
Atlanta	0.018876
Palmdale	0.000344
Islip	0.001879
...	...
Wichita	0.005385
Kansas City	0.011362
Melbourne	0.000746
Duluth	0.000718
Bangor	0.002427
Lafayette	0.001162
Mosinee	0.001586
Lewisburg	0.000411
Rock Springs	0.000700
Corpus Christi	0.001162
Houston	0.016778
Pellston	0.000695
Tallahassee	0.002187
Bethel	0.000115
Roswell	0.000385
Philadelphia	0.012434
Tupelo	0.000411

Loading...

Tupelo	0.000411
Durango	0.001197
Reno	0.005229
Pittsburgh	0.008777

Loading...

Calculating a ratio: the worst performing airports relative to their performance in terms of the network (which help us find how "weak" those links are):

Honolulu	0.000548

```
for names in sizes_ar.keys():
    sizes_ar[names] = float(ar_delays[ar_delays.destin_city==names]['ArrDelay'] / sizes_ar[names])

for names in sizes_dep.keys():
    #if names !='Pierre':
    sizes_dep[names] = float(dep_delays[dep_delays.origin_city==names]['DepDelay'] / sizes_dep[names])
```

Oakland	0.007017

```
# Save in separate csvs for prettier plotting
ar_df = pd.DataFrame([]).from_dict(sizes_ar,orient='index').reset_index()
ar_df.columns = ['destin_city','ind']
dep_df = pd.DataFrame([]).from_dict(sizes_dep,orient='index').reset_index()
dep_df.columns = ['origin_city','ind']
ar_df["destin_lat"]   = ar_df["destin_city"].map(airports['lat'])
ar_df["destin_long"]  = ar_df["destin_city"].map(airports['long'])
dep_df["origin_lat"]  = dep_df["origin_city"].map(airports['lat'])
dep_df["origin_long"] = dep_df["origin_city"].map(airports['long'])
dep_df.to_csv('Departure_vulnerability.csv')
ar_df.to_csv('Arrival_vulnerability.csv')


ar_df.sort_values('ind',ascending=False).head(10)
```

	destin_city	ind	destin_lat	destin_long
28	Atlanta	0.160344	45.000008	-84.133337
161	Salt Lake City	0.128662	40.619540	-111.992886
56	Minneapolis	0.127824	44.880547	-93.216922
133	Dallas	0.126783	32.680861	-96.868194
167	Detroit	0.101268	42.237928	-83.530409
74	Denver	0.076429	39.785250	-104.543139
165	Chicago	0.062064	41.979595	-87.904464
242	Houston	0.059642	29.508361	-95.051333
229	Anchorage	0.057610	61.214379	-149.846161
212	Phoenix	0.055963	33.434167	-112.008056

```
dep_df.sort_values('ind',ascending=False).head(10)
```

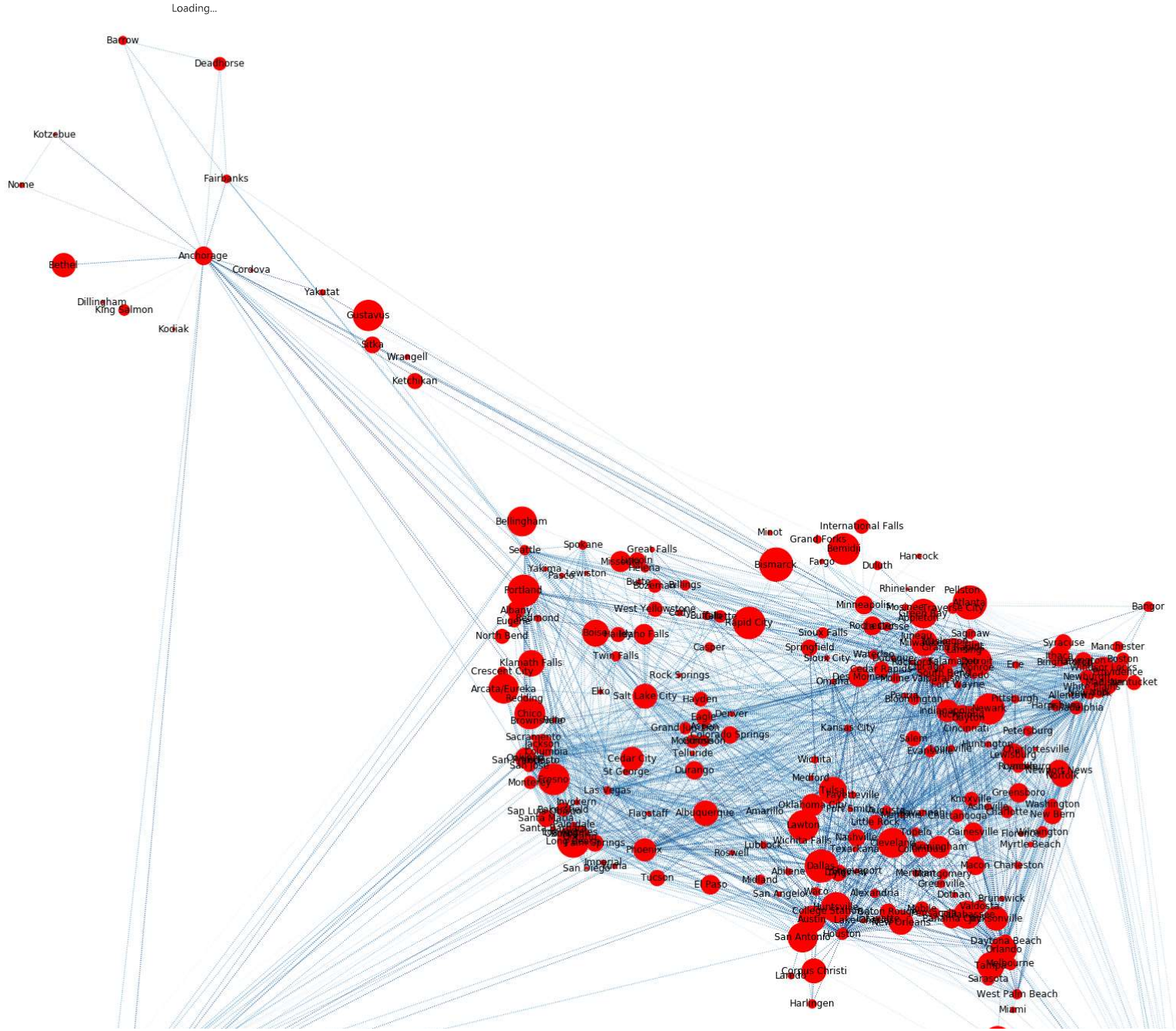
	origin_city	ind	origin_lat	origin_long
28	Atlanta	0.160344	45.000008	-84.133337
161	Salt Lake City	0.128662	40.619540	-111.992886
56	Minneapolis	0.127824	44.880547	-93.216922
133	Dallas	0.126783	32.680861	-96.868194
167	Detroit	0.101268	42.237928	-83.530409
74	Denver	0.076429	39.785250	-104.543139
165	Chicago	0.062064	41.979595	-87.904464
242	Houston	0.059642	29.508361	-95.051333
229	Anchorage	0.057610	61.214379	-149.846161
212	Phoenix	0.055963	33.434167	-112.008056

And, plotting the "vulnerability index"...

```
plt.figure(figsize=(30,30))
plt.title('Departure vulnerability index',fontsize=35)
```

```
nx.draw(G,pos=pos['coord'],with_labels=True,
        node_size=10000E1*np.array(sizes_dep.values()),style='dotted', edge_color=range(G.number_of_edges()), edge_cmap=plt.cm.Blues,node_color='r')
```

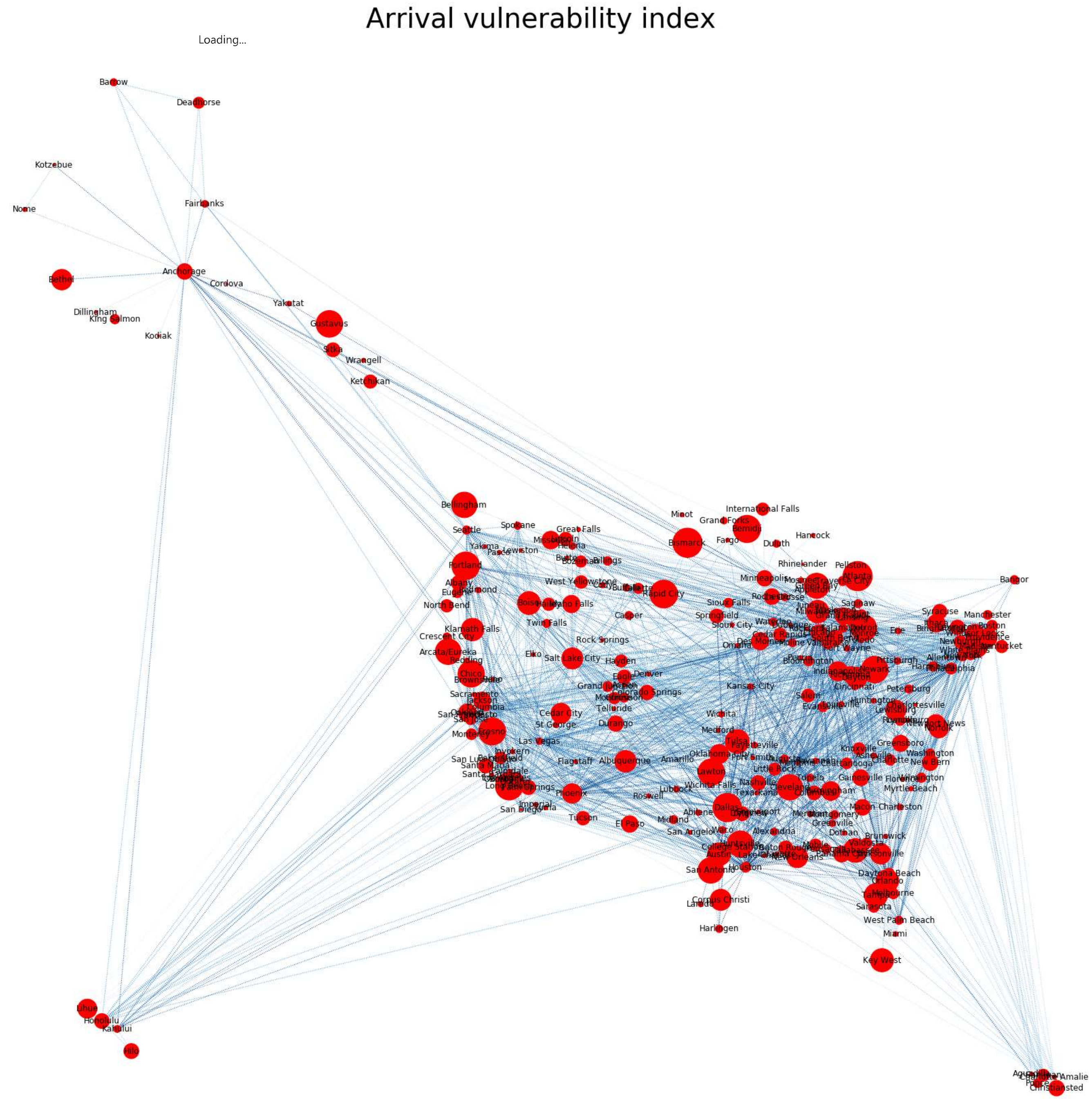
Departure vulnerability index




```
plt.figure(figsize=(30,30))
```

```
plt.figure(figsize=(50,50))
plt.title('Arrival vulnerability index',fontsize=40)
nx.draw(G,pos=pos['coord'],with_labels=True,
        node_size=10000E1*np.array(sizes_ar.values()),style='dotted', edge_color=range(G.number_of_edges()), edge_cmap=plt.cm.Blues,node_color='r')
```

Loading...



```
data = data[[ u'ActualElapsedTime','UniqueCarrier' ,u'CRSElapsedTime', u'origin_city', u'destin_city',u'origin_lat', u'origin_long', u'destin_lat', u'destin_long', u'coord']]
data.head()
```

	ActualElapsedTime	UniqueCarrier	CRSElapsedTime	origin_city	destin_city	origin_lat	origin_long	destin_lat	destin_long	coord
0	128.0	WN	150.0	Washington	Tampa	35.570468	-77.049813	28.013984	-82.345279	(-77.04981306, 35.57046806)
1	128.0	WN	Loading... 145.0	Washington	Tampa	35.570468	-77.049813	28.013984	-82.345279	(-77.04981306, 35.57046806)
2	96.0	WN	90.0	Indianapolis	BWI Airport	39.935203	-86.044953	NaN	NaN	(-86.04495333, 39.9352025)
3	88.0	WN	90.0	Indianapolis	BWI Airport	39.935203	-86.044953	NaN	NaN	(-86.04495333, 39.9352025)
4	90.0	WN	90.0	Indianapolis	BWI Airport	39.935203	-86.044953	NaN	NaN	(-86.04495333, 39.9352025)

Performance vs. schedule

[ ] ↳ *4 cells hidden*