HDDT analysis - Locations

An analysis of the locations associated with members of the CEDA 1830 -1870

The graph illustration above shows all of the location data in the database rendered by gexf (see iterative sections 1 and 2)

Introduction

1881 members of the community are recorded as members of 81 locations and each of these locations constitutes a locality where meetings between members may have taken place, equally they may also be places where members might meet up infrequently or informally. The visual analysis of connectivity between members at individual locations and between locations indicates the extent that the community is geographically connected. The 1881 make up 61% of the entire community.

The above graph shows the 1881 distributed geographically with the connectivity between them reflected in those members who have been associated with more than one location.

In [59]:
```
# First we call up the python packages we need to perform the analysis:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(20, 10))
from operator import itemgetter
import networkx as nx
from networkx.algorithms import community #This part of networkx, for community detection, needs to be imported separ
import nbconvert
import csv
```

Iterative Section 1 - (This is an iterative workbook)

As can be seen in the illustrative graph above which has been produced in Gephi to provide an initial overview of the data and its distribution, the graph might be made more meaningful if it did not include locations sparsely populated.

The code cell below and the code cells in the section Iterative Section 2 (below) have been designed so that a second run through the workbook can subsequently be made where the second run uses data that excludes low populated areas.

In [60]:

```
# Second we call up the csv files generated from the SQL database that contain information about
# locations and the community members associated with locations. As well as enabling the locations to be listed
# we produce a 'node_names' file and a tuples file of edges_attributes to generate the files need to
# produce GefX files for Gephi.

# We can run the code cell twice, first with all data and once all data has been examined
# and a decision made to exclude 'noise' the code block can be run again with newly generated
# csv files that exclude low populated locations.

location = pd.read_csv ('location_202107221303.csv')

# Use these csv files in the 'with open' statements below to generate locations.gexf
names = pd.read_csv ('vw_3_location_names_1_2_202107221500.csv')# For nodes csv
tuples = pd.read_csv ('vw_2_location_membership_xid_202107221527.csv')# For edges.csv

# Use these csv files in the 'with open' statements below to generate locations_10.gexf
location_10_names = pd.read_csv ('vw_location_10_names_1_2_202107231036.csv') # For nodes csv
location_10_tuples = pd.read_csv ('vw_location_10_202107231013.csv') # For edges.csv


with open('vw_2_location_membership_xid_202107221527.csv', 'r') as nodecsv: # Open the Nodes csv file
    nodereader = csv.reader(nodecsv) # Read the csv
    nodes = [n for n in nodereader][1:]# Retrieve the data (using Python list comprhension and list slicing
                                       # to remove the header row
    node_names = [n[0] for n in nodes] # Get a list of only the node names

with open('vw_location_10_202107231013.csv', 'r') as edgecsv: # Open the file
    edgereader = csv.reader(edgecsv) # Read the csv
    edge_list = list(edgereader) # Convert to list, so can iterate below in for loop

    # Create empty arrays to store edge data and edge attribute data
    edges = []
    edges_attributes = []

    # Fill the arrays with data from CSV
    for e in edge_list[1:]:
        edges.append(tuple(e[0:2])) # Get the first 2 columns (source, target) and add to array
        # not used this time. edges_attributes.append(tuple(e[2:4]))
        # Get the 3rd and 4th columns (first_year, last_year) and add to array

edge_names = [e[0] for e in edges] # Get a list of only the edge names
```

We begin by listing out and validating all of the location data in the database

In [61]:
```
# List out the locations to be analysed

location
```

Out[61]:

	id	name	notes
0	1	London	NaN
1	3	country	NaN
2	4	Africa	NaN
3	5	America	NaN
4	6	Scotland	NaN
...	...	...	...
78	80	Madagascar	NaN
79	81	Ecuador	NaN
80	82	Seychelles	NaN
81	83	Panama	NaN
82	84	Armenia	NaN

83 rows × 3 columns

In [62]:
```
# List out the community members who have been associated with at least one location.

names
```

Out[62]:

	Name
0	A Mackintosh Shaw
1	A , jun Ramsay
2	A A Stewart

	Name
3	A B Stark
4	A C Brebner
...	...
1876	x Tumangung of Jehore
1877	x University Library
1878	x Wagstaff
1879	x Wienecke
1880	x Zohrab

In [63]:
```
# Finally list out the tuples of members and locations
# (Note - some members are associated with more that one location)

tuples
```

Out[63]:

	Source	Target
0	Arthur William A Beckett	London
1	Andrew Mercer Adam	country
2	H R Adam	Africa
3	Henry John Adams	London
4	William (2) Adams	London
...	...	...
2056	James A Youl	London
2057	Robert Younge	country
2058	Arthur de Zeltner	Panama
2059	x Zohrab	Armenia
2060	x Zohrab	Turkey

2061 rows × 2 columns

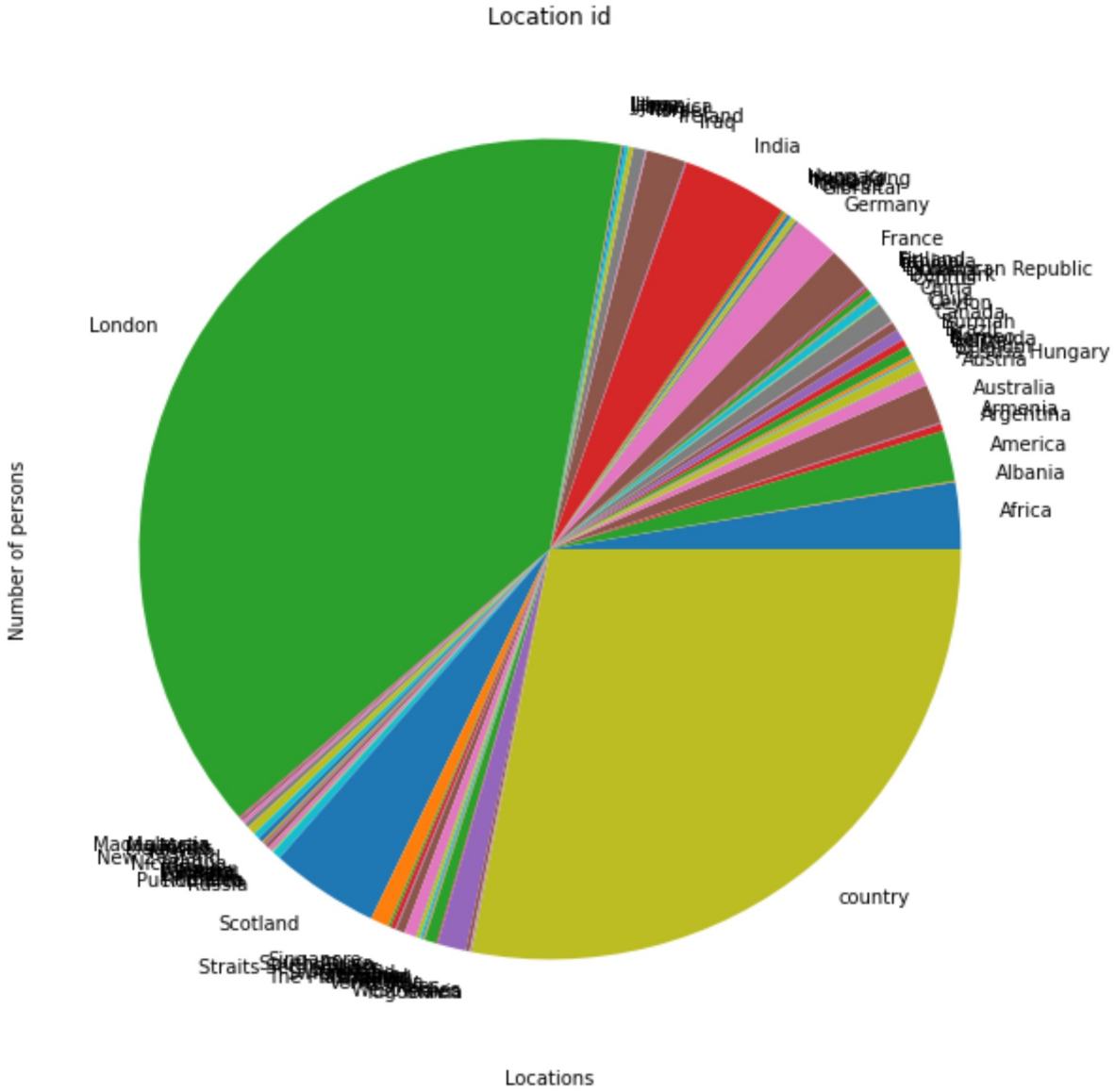
Use pyplot to make an inital visualisation of the data

We can see that many locations are thinly populated.

We can also see that locations which might be characterised as parts of the British Empire are well represented with locations also found in prominent cities in other colonialist centres (the US, France, Germany).

We should be cautious about the location 'country' because this is not a location, its is a catch all cetegory for the member networks in England but not in London.

In [64]:
```
tuples.groupby('Target')['Source'].nunique().plot(kind='pie')
plt.title ("Location id")
plt.xlabel ("Locations")
plt.ylabel ("Number of persons")
plt.show()
```



Iterative Section 2 - prepare the data for rendering as a graph in Gephi

Caution - this section depends on the selections made under 'Iterative Section 1' above

If the initial analysis suggests that a more insightful visualisation might be made by refing the data to be analysed, return to the database and make a new Nodes (Names) csv file and a new Tuples csv file.

Return to Iterative Section 1 codeblock in the workbook and replace the csv files in the 'with open' code lines.

Set the nx.write_gexf (xxx.gexf) xxx statement to a new file name.

Then run all code blocks again and make a more insightful gexf file for Gephi.

Ensure that the statement nx.write_gexf(G, 'locations.gexf') in the last code cell in this section points to the desired output file for Gephi. Failure to set this value correctly will result in the previously generated .gexf file being overwritten

In [65]:
```
print("Nodes length: ", len(node_names))
print("Edges length: ", len(edges))
# not used this time. print("Edges attributes length: ", len(edges_attributes)) # This should be the same length as e
```

```
Nodes length:  2061
Edges length:  1790
```

In [66]:
```
# First check that the data is correctly formatted

print("First 5 nodes:", node_names[0:5])
print("First 5 edges:", edges[0:5])
# not used this time. print("First 5 edges attributes:", edges_attributes[0:5])

# The output will appear below this code cell.
```

```
First 5 nodes: ['Arthur William A Beckett', 'Andrew Mercer Adam', 'H R Adam', 'Henry John Adams', 'William (2) Adams
']
First 5 edges: [('Arthur William A Beckett', 'London'), ('Andrew Mercer Adam', 'country'), ('H R Adam', 'Africa'), ('
Henry John Adams', 'London'), ('William (2) Adams', 'London')]
```

In [67]:
```
# We use NetworkX to build the graph data into a table

G = nx.Graph()
G.add_nodes_from(node_names)
G.add_edges_from(edges)
print(nx.info(G))
```

```
Name:
Type: Graph
Number of nodes: 1808
Number of edges: 1790
```

```
Average degree:   1.9801
```

In [68]:
```
# Finally we can write a gexf file which will be placed in the root directory.
# We can then open the file in Gephi and visualise the network.

nx.write_gexf(G, 'locations.gexf')
```

Stage 2 - Locations analysis with 'noise' removed (low populated locations excluded).

We now re-run the code to generate a new gexf file for gephi. We use the refined pair of nodes (Names) and Tuples files generated in the SQL database that include only the top 10 locations.

In [69]:
```
location_10_names
```

Out[69]:

	Name
0	A Mackintosh Shaw
1	A , jun Ramsay
2	A A Stewart
3	A B Stark
4	A C Brebner
...	...
1617	x Strangford
1618	x Tinsley
1619	x University Library
1620	x Wagstaff
1621	x Wienecke

1622 rows × 1 columns

In [70]:
```
location_10_tuples
```

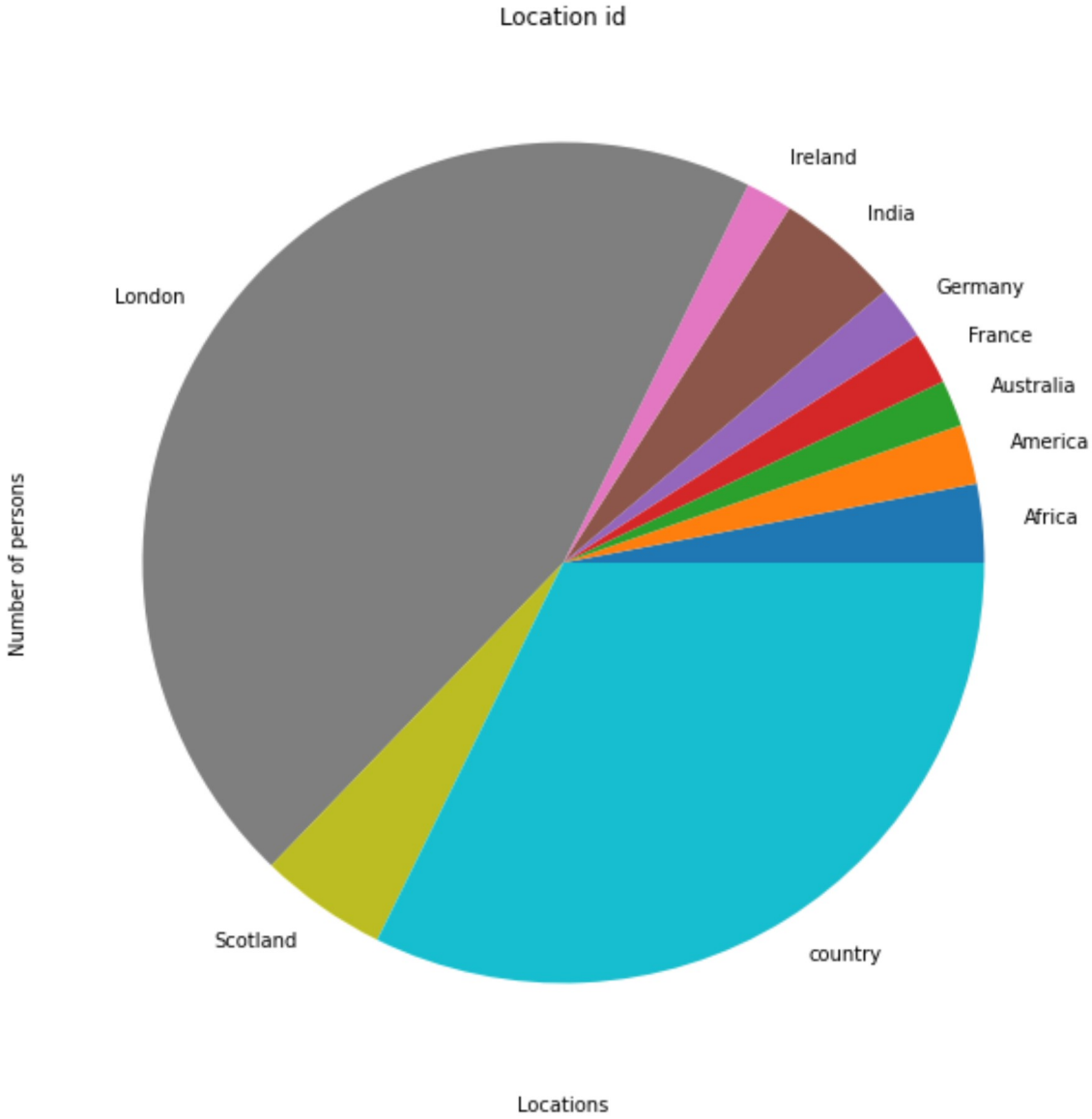
Out[70]:

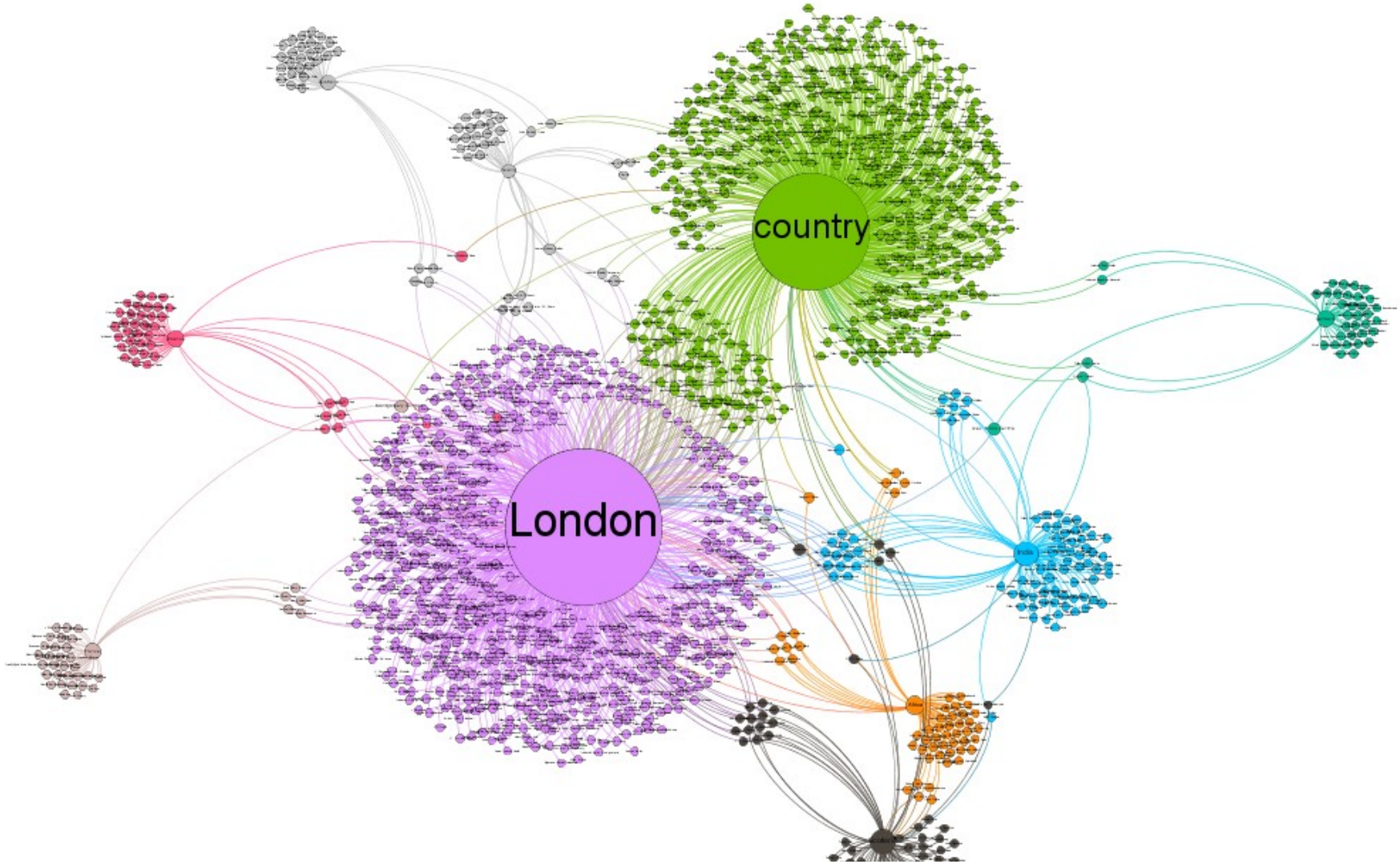
	Source	Target
0	Arthur William A Beckett	London
1	Andrew Mercer Adam	country
2	H R Adam	Africa
3	Henry John Adams	London
4	William (2) Adams	London
...	...	...
1785	Ashton Yates	country
1786	James Yearsley	country
1787	Stephen Yeldham	London
1788	James A Youl	London
1789	Robert Younge	country

1790 rows × 2 columns

We now have a graph of the dataset that can be analysed more easily

In [71]:
```
location_10_tuples.groupby('Target')['Source'].nunique().plot(kind='pie')
plt.title ("Location id")
plt.xlabel ("Locations")
plt.ylabel ("Number of persons")
plt.show()
```


Location id
Ireland
India
Germany
France
Australia
America
Africa
London
Scotland
country
Number of persons
Locations


country
London

We can see that London and 'country'(sic) are the most populated locations. Because the 'country' location is an aggregate (and not a specific location) we can think of London and 'country' as a twin centre. Within the twin centre we can see the members of both London and 'country' locations and that the members of each are highly networked. We can also see that the London location contains many members who have no association with any other group (including 'country'). London 1830 - 1870, was densely populated and so it is possible that members of the London location had other modes of association. Because the 'country' location is an aggregate we cannot make the same analysis to the same extent, it is possible that many members in (say) Newcastle had no association with other members in (say) Bristol. We can see the large group of members who were members of both London and 'country' locations. It is highly likely that these members served as conduits of communication and group cohesion. It is interesting to note that only 3 members of this London and 'country' group were members of groups outside of the twin centre.

Eight other location each have a membership of around 30 members (we can call these the satellites), all of the satellite groups relate directly to the twin centre with very few members associated with more that one satellite location.

Australia and Ireland have associations with both London and 'country'. The German location is most closely associated with the 'country' group. All of the other locations are strongly associated with the London location.

Germany (far right) is the location least associated with London. Alex Nidda Genthe is the only member from Germany who is also a member of the London location. Friedrich Max Muller, Frederick Augustus Haverick and Gustav Oppert each network with 'country' members. William Wilson Hunter is the only 'country' member who also appears in the Germany location. He and Gustav oppert also have a location connection with India.

In [ ]: