Clustering Migration Data via Singular Value Decomposition

Joan Wang, Max Del Giudice

December 15, 2012

Abstract

Given an $m \times n$ matrix containing migration data between various countries, how may one effectively partition the data into k neighborhoods, each of which denotes a meaningful relationship between its component countries? Moreover, how does one effectively choose k? The singular value decomposition (SVD) of a matrix provides an answer to both questions, allowing one to bi-cluster datapoints based on the resulting left and right singular-vector matrices. Using the SVD, we draw a number of conclusions regarding the inflow and outflow of migrants among countries, and close by assessing the efficacy and accuracy of this method in segmenting asymmetric data sets.

Contents

List of Figures

List of Tables

1 Introduction

How are countries characterized by their migration statistics? In general, people move abroad in search of work opportunities and better living standards. However, several different theories address the complexities of migration in the 21st century, including: the dual labor market theory, the relative deprivation theory, and the world systems theory, among many others. Unfortunately, testing these theories with raw data is next to impossible, suggesting a mathematical approach as a promising alternative. Beginning with a weighted graph representation of our migration data, spectral clustering provides a method for forming vector representations of our vertices in Euclidean n-space. Using traditional clustering algorithms (such as the k-means algorithm), we are able to group the vector representations into meaningful clusters. We hope to challenge the premises of these migration theories with our results, and more importantly, assess the viability of SVD clustering in analyzing migration networks.

Section 2 provides the economic and mathematical preliminaries for the paper. Section 3 states some mathematical theories and motivations behind the research topic. Section 4 is devoted to a step-by-step introduction to spectral clustering for undirected graphs ending with motivation for the SVD approach. Section 5 presents the SVD in detail by applying the theory of clustering based on the SVD to the migration dataset. Section 6 presents and explains results of clustering via the SVD. Section 7 summarizes our mathematical and social findings.

2 Preliminaries

2.1 Mathematical Notation

Let $G = (V, \mathbf{W})$ denote an edge-weighted graph on n vertices, where $|V| = n$ is the vertex set, and $\mathbf{W}$ is the weight matrix. The following notation will be used freely throughout the paper:

- $d_i := \sum_{j=1}^{n} w_{ij}$ $(i = 1, \ldots, n)$ is the generalized degree of i,
- $\mathbf{d} := (d_1, \ldots, d_n)^T$ is the degree vector, $\sqrt{\mathbf{d}} := (\sqrt{d_1}, \ldots, \sqrt{d_n})^T$
- $\mathbf{D} := \text{diag}(d_1, \ldots, d_n)$ is the degree matrix.
- $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Graph Laplacian of G
- $\mathbf{L}_D = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is the normalized Laplacian of G

Without loss of generality, we assume $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} = 1$.

2.2 Economic Background

We are going to examine the clustering results of the migration dataset in relation to three main theories of migration: the dual labor market theory, the relative deprivation theory, and the world systems theory. We hypothesize that at least one of these theories would be confirmed with our clustering results.

The dual labor market theory claims that migration is mainly caused by the pull factors in more developed countries. Pull factors are traits that attract one to another country while push factors are adverse traits of the country one lives in. This theory assumes that labor markets in more developed countries consists of the primary segment (high-skilled labor) and the secondary segment (low-skilled labor). If a country has a shortage of the secondary labor force, it would push wages up in an attempt to attract secondary workers. This creates a strong pull factor for people to migrate.

The relative deprivation theory states that high income inequality in one's country is a strong push factor and the main reason for migration.

The world systems theory looks at migration from a global perspective. A part of the theory argues that even after decolonization, the economic dependence of former colonies still remains on mother countries. Therefore, if there exists migration between countries that are geographically far apart, this theory might be able to explain that phenomenon.

2.3 Data

There are two separate migration datasets collected from LABOSTA, an International Labor Office database operated by International Labor Organization (ILO) Department of Statistics. The ILO Department of Statistics is the "focal point within the UN system for labor statistics"[1]. Both datasets are downloaded as csv files with available data from 140 countries between the years 1986 and 2005.

The "laborstaM6.csv" raw dataset contains inflows of migrants of each country by sex and by country of origin. The "laborstaMB.csv" raw dataset contains outflows of nationals of each country by sex and by country of destination. These two datasets should be regarded as two completely separate datasets since there is actually not a one-to-one correspondence between the two. Furthermore, since both datasets contain essentially the same information (country origin, country of destination, and the amount of people migrating from the former to the latter), they can be analyzed separately but parallel (with the same methods).

This paper only uses data from "laborstaM6.csv" and from the year 2006 for the most updated statistics and creates its own numbering system of the countries in order to easily label the vertices of the directed graphs. We drop the insignificant migration records based on the individual amount of migration compared to the total amount of migration in 2006 (if the relative weights fall below 1.0e-5). The edited dataset is contained in Sheet 1 of a separate Excel workbook titled "lnflows-nb". Sheet 2 of "Inflows-nb" contains the population weights of the countries of destination while Sheet 3 of "Inflows-nb" contains the population weights of the countries of origin. The population of these countries in 2006 are obtained from the United Nations Statistics Division[7]. The entries with missing data are filled in with results from the World Bank[9]. For all three worksheets of "Inflows-nb," see Appendix A.

3 Theoretical Motivation and Results

3.1 Multiway Cut Problems

Intuitively, when clustering a large graph we want to group vertices by their relative similarities. Ideally, the sum of the edge weights between clusters will be very low, whereas the sum of the edgeweights within clusters will be high. Normalized spectral clustering is a viable method for accomplishing such a goal, and we may derive the process by examining graph cut problems.

Let $G = (V, \mathbf{W})$ be an edgeweighted non-directed graph. To formalize the above notion, we define the following function,

$$\text{cut}(C_1, ..., C_k) := \sum_{i<j} W(C_i, C_j)$$

where $\{C_1, ..., C_k\}$ represents a partition of the graph into k clusters, and $W(C_i, C_j)$ is the sum of the edge weights between cluster i and j. It is clear that we wish to minimize this sum. Unfortunately, minimizing the above sum often results in single vertex clusters, and such neighborhoods give us no information. In a sense, we want to ensure that the clusters are reasonably sized by introducing some sort of normalizing factor into our sum. There are multiple approaches to take, but we will only examine one such method. Define a new objective function,

$$\text{Ncut}(C_1, ..., C_k) := \sum_{i<j} \left(\frac{1}{\text{vol}(C_i)} + \frac{1}{\text{vol}(C_j)} \right) W(C_i, C_j) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{\text{vol}(C_i)}$$

where $\text{vol}(C_i)$ is the sum of the degrees in cluster C_i, and $\overline{C_i}$ is the complement of cluster C_i. Minimizing Ncut is NP complete, but normalized spectral clustering can be derived as a method for solving a relaxed version of this problem. A derivation can be found in [Luxburg]

When clustering a graph into k clusters, we are looking to find k-dimensional representative $\mathbf{r}_1, ..., \mathbf{r}_n$ such that they minimize the following objective function:

$$Q_k := \sum_{i<j} w_{ij} ||\mathbf{r}_i - \mathbf{r}_j||^2 \quad \text{subject to} \quad \sum_{i=1}^{n} \mathbf{r}_i \mathbf{r}_i^T = \mathbf{I}_k$$

Minimizing the above function (subject to the given constraint) yields a placement that forces vertices with large edge-weights to be close to one another. However, it would be beneficial to rewrite the objective function in a more illuminating manner. Before doing so, we define $\mathbf{X}$ as the $n \times k$ matrix with rows $\mathbf{r}_1^T, ..., \mathbf{r}_n^T$. Defining $\mathbf{x}_1, ..., \mathbf{x}_k \in \mathbb{R}^n$ as the columns of $\mathbf{X}$, we may write $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_k)$. The constraint given with our objective function can now be reformulated as $\mathbf{X}^T\mathbf{X} = \mathbf{I}_k$. With this machinary in place, we rewrite the objective function as,

$$\begin{aligned} Q_k &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} ||\mathbf{r}_i - \mathbf{r}_j||^2 = \sum_{i=1}^{n} d_i ||\mathbf{r}_i||^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \mathbf{r}_i^T \mathbf{r}_j \\ &= \sum_{l=1}^{k} \mathbf{x}_l^T (\mathbf{D} - \mathbf{W}) \mathbf{x}_l = \text{tr}(\mathbf{X}^T \mathbf{L} \mathbf{X}) \end{aligned}$$

The next results provide a solution to the above minimization problem.

Lemma 3.1.1: Let $G = (V, \mathbf{W})$ be an undirected edge-weighted graph with Laplacian $\mathbf{L}$. Furthermore, let $0 = \mu_0 \leq \mu_1 \leq \cdots \leq \mu_{n-1}$. Let $k < n$ be an integer such that $\mu_{k-1} < \mu_k$. Using the notation developed above, we have,

$$\min_{\sum_{i=1}^{n} \mathbf{r}_i \mathbf{r}_i^T = \mathbf{I}_k} Q_k = \min_{\mathbf{X}^T\mathbf{X} = \mathbf{I}_k} \text{tr}(\mathbf{X}^T \mathbf{L} \mathbf{X}) = \sum_{i=0}^{k-1} \mu_i$$

Proof. Follows from the Rayleigh-Ritz Theorem. □

Lemma 3.1.2: Let $G = (V, \mathbf{W})$ be an undirected edge-weighted graph with normalized Laplacian $\mathbf{L}_D$. Define $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1} \leq 2$ to be the eigenvalues of $\mathbf{L}_D$. Let $k < n$

be an integer such that $\lambda_{k-1} < \lambda_k$. Using the notation developed above, we have,

$$\min_{\sum_{i=1}^{n} d_i \mathbf{r}_i \mathbf{r}_i^T = \mathbf{I}_k} Q_k = \min_{\mathbf{X}^T\mathbf{D}\mathbf{X}=\mathbf{I}_k} \operatorname{tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) = \sum_{i=0}^{k-1} \lambda_i$$

Proof. Note that $(\mathbf{D}^{1/2}\mathbf{X})^T(\mathbf{D}^{1/2}\mathbf{X}) = \mathbf{X}^T\mathbf{D}\mathbf{X} = \mathbf{I}_k$. Let $\mathbf{A} = \mathbf{D}^{1/2}\mathbf{X}$, and note the following

$$\min_{\mathbf{A}^T\mathbf{A}=\mathbf{I}_k} \operatorname{tr}(\mathbf{A}^T\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{A}) = \min_{\mathbf{A}^T\mathbf{A}=\mathbf{I}_k} \operatorname{tr}(\mathbf{A}^T\mathbf{L}_D\mathbf{A})$$

We may now apply lemma 1, and our result follows. □

With these two preliminary results, we are ready to develop the link between spectra and the Normalized cut problem.

Theorem 3.1.3: Let $G = (V, \mathbf{W})$ be an undirected, edge-weighted graph with normalized Laplacian $\mathbf{L}_D$. Furthermore, let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq 2$ be the eigenvalues of $\mathbf{L}_D$. Choosing some k such that $k < n$ and $\lambda_{k-1} < \lambda_k$, we have that,

$$\operatorname{Ncut}(A_1, ..., A_k) \geq \sum_{i=1}^{k-1} \lambda_i$$

Proof. Let $k > 2$, and define the indicator vectors $\mathbf{x}_j = (x_{1,j}, ..., x_{n,j})^T$ by

$$x_{i,j} = \begin{cases} 1/\sqrt{\operatorname{vol}(A_j)} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathbf{X}$ be the matrix containing these $\mathbf{x}_i$ as columns. We make the following observations:

$$\mathbf{X}^T\mathbf{D}\mathbf{X} = \mathbf{I}_k$$

$$x_i^T\mathbf{D}x_i = \frac{\operatorname{cut}(A_i, \overline{A}_i)}{\operatorname{vol}(A_i)}$$

Using the machinery and notation developed above, we may reformulate the problem of minimizing Ncut as follows. Again, let $\mathbf{A} = \mathbf{D}^{1/2}\mathbf{H}$:

$$\begin{aligned} \min \operatorname{Ncut}(A_1, ..., A_k) &= \min_{\mathbf{X}^T\mathbf{D}\mathbf{X}=\mathbf{I}_k} \operatorname{tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \\ &= \min_{\mathbf{A}^T\mathbf{A}=\mathbf{I}_k} \operatorname{tr}(\mathbf{A}^T\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{A}) \\ &= \min_{\mathbf{A}^T\mathbf{A}=\mathbf{I}_k} \operatorname{tr}(\mathbf{A}^T\mathbf{L}_D\mathbf{A}^T) \end{aligned}$$

As shown in Lemma 2, this minimization problem is solved by $\sum_{i=1}^{k-1} \lambda_i$, where $\{\lambda_1, ..., \lambda_{k-1}\}$ are the eigenvalues of the normalized Laplacian $\mathbf{L}_D$. Therefore we have,

$$\operatorname{Ncut}(A_1, ..., A_k) \geq \min \operatorname{Ncut}(A_1, ..., A_k) \geq \sum_{i=1}^{k-1} \lambda_i$$

□

3.2 The Isoperimetric Number and Associated Bounds

Definition 3.2.1: Let $G = (V, \mathbf{W})$ be an edge-weighted graph with generalized degrees $d_1, ...d_n$ and suppose that $\sum_{i=1}^{n} d_i = 1$. The *isoperimetric number* (or *Cheegar constant*) of G is

$$h(G) = \min_{\substack{U \subset V \\ \text{Vol}(U) \leq \frac{1}{2}}} \frac{w(U, \overline{U})}{\text{vol}(U)}$$

The following theorem places a lower bound on the isoperimetric number of a graph G.

Theorem 3.2.2: Let $G = (V, \mathbf{W})$ be a connected edge-weighted graph with isoperimetric number $h(G)$, and let λ_1 denote the smallest positive eigenvalue of its normalized Laplacian $\mathbf{L_D}$. Then,

$$\frac{\lambda_1}{2} \leq h(G)$$

Proof. Let $G = (V, \mathbf{W})$ be an edge-weighted graph on n vertices. We note the following preliminary result,

$$\lambda_1 = \min_{\substack{\sum_{i=1}^{n} d_i r_i = 0 \\ \sum_{i=1}^{n} d_i r_i^2 = 1}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}(r_i - r_j)^2 = \min_{\sum_{i=1}^{n} d_i r_i = 0} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}(r_i - r_j)^2}{\sum_{i=1}^{n} d_i r_i^2}$$

Let A denote a vertex subset of G over which the minimum of definition 3.2.1 is attained. Furthermore, we represent the vertices of G as follows.

$$r_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\text{vol}(A)} & \text{if } i \notin A \end{cases}$$

Then the following computation holds:

$$\begin{aligned}
\lambda_1 &\leq \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (r_i - r_j)^2 w_{ij}}{\sum_{i=1}^{n} d_i r_i^2} = \frac{\sum_{i \in A} \sum_{j \in \overline{A}} \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(\overline{A})} \right)^2 w_{ij}}{\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(\overline{A})}} \\
&= \frac{\text{vol}(A) + \text{vol}(\overline{A})}{\text{vol}(A)\text{vol}(\overline{A})} \sum_{i \in A} \sum_{j \in \overline{A}} w_{ij} = \frac{1}{\text{vol}(A)\text{vol}(\overline{A})} \cdot w(A, \overline{A}) \\
&\leq 2 \frac{w(A, \overline{A})}{\text{vol}(A)} = 2h(G)
\end{aligned}$$

Because $\lambda_1 \leq 2h(G)$, we have $\frac{\lambda_1}{2} \leq h(G)$. □

We note that there also exists an upper-bound for the isoperimetric number, $h(G) \leq \min\{1, \sqrt{2\lambda_1}\}$. We omit the proof as it is quite involved, though The interested reader may refer to [3]. Together, these two bounds comprise the *Cheeger Inequality.* While we gave a full derivation for the lower bound, we note there is an alternate method using our prior results.

Proof. Let

$$f_k(G) = \min \text{Ncut}(A_1, ..., A_k), \quad \{A_1, A_2, ..., A_k\} \text{ is a k-clustering of } G$$

Furthermore, we have that

$$\text{Ncut}(A, \overline{A}) = \frac{\text{cut}(A, \overline{A})}{\text{vol}(A)} + \frac{\text{cut}(\overline{A}, A)}{\text{vol}(\overline{A})} = \frac{\text{cut}(A, \overline{A})}{\text{vol}(A)\text{vol}(\overline{A})} \leq \frac{2\text{cut}(A, \overline{A})}{\text{vol}(A)}$$

The last inequality follows because we assume that $\text{vol}(A) \leq 1/2$. Using theorem 3.1.3, the Cheegar inequality immediately falls out,

$$\lambda_1 \leq f_2(G) \leq 2h(G)$$

From which it is apparent that $\lambda_1/2 \leq h(G)$. □

4 Spectral Clustering for Undirected Graphs

4.1 Graph Laplacians and Representation

If G is connected ($\mathbf{W}$ is irreducible), then the normalized Laplacian, $\mathbf{L_D}$, has 0 as a single eigenvalue with corresponding unit-norm eigenvector $\sqrt{\mathbf{d}}$. More generally, if G is an undirected graph with non-negative weights, then the multiplicity s of the eigenvalue 0 of $\mathbf{L_D}$ equals the number of connected components in the graph. We focus on relaxing the Ncut, which leads to normalized spectral clustering. As a result, we will use the normalized Laplacian from now on.

G is connected, $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{n-1} \leq 2$ are the eigenvalues of $\mathbf{L_D}$ with corresponding unit-norm, pairwise orthogonal eigenvectors $\mathbf{u_0} = \sqrt{\mathbf{d}}, \mathbf{u_1}, \ldots, \mathbf{u_{n-1}}$. Only bipartite graphs have 2 as an eigenvalue; all other graphs have eigenvalues strictly less than 2.

We should be able to find a sufficiently large gap in this ordered sequence of eigenvalues: $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{k-1} < \lambda_k \leq \cdots \leq 2$. A sufficiently large gap between λ_{k-1} and λ_k means that this gap is the last remarkable difference and all the gaps after this one are insignificant. It does not necessarily mean that the gaps before are smaller than the one between λ_{k-1} and λ_k. The number of eigenvalues before this gap would be the number of desired clusters, denoted by k.

Then, we construct the optimal representation of the vertices in a form of the matrix $\mathbf{X_{n\times k}} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{u_0}, \ldots, \mathbf{D}^{-\frac{1}{2}}\mathbf{u_{k-1}})$, where $\mathbf{u_i}$ is the eigenvector belonging to the ith eigenvalue of $\mathbf{L_D}$. Note that $\mathbf{x_0} = \mathbf{D}^{-\frac{1}{2}}\mathbf{u_0} = 1$, which does not give us any nontrivial information about the vertices. Therefore, we disregard $\mathbf{x_0}$, the first column of $\mathbf{X}$. We write $\mathbf{r_1}, \ldots, \mathbf{r_n}$ as the row vectors of $\mathbf{X}$ without the first component. Now, these vectors with $k-1$ dimensions are called the vector representations where $\mathbf{r_i}$ is the vector representation of the ith vertex.

This step is the point of spectral clustering since this representation makes it possible to place the vertices in a finite-dimensional space. We can then use traditional clustering algorithms, such as the k-means algorithm, to put each vertex into its appropriate cluster.

We are able to visually represent the vertices of a graph with these vector representatives in two or three dimensions. For more than two clusters ($k > 2$): a 2-D representation plot would have n points with the ith point having the coordinates (x_{i1}, x_{i2}). The first and second components of the vector $\mathbf{r_i}$ now represent the ith vertex in the plot. A 3-D representation plot would have n points with the ith point having the coordinates (x_{i1}, x_{i2}, x_{i3}). In this case, the first, second, and third components of $\mathbf{r_i}$ altogether represent the ith vertex in the plot.

4.2 Clustering

4.2.1 Simple k-means

The k-means algorithm starts with a set of k clusters, and computes the centers of the clusters. Then, it puts each vertex in a cluster if the squared distance between the representative vector of the vertex and the center of the cluster is the smallest out of all other distances. Once all vertices go through this process of verification and possible relocation, the centers of the new clusters are recalculated, and then more vertices are relocated, so on and so forth. The whole procedure is repeated until the vertices eventually converge and stay in their respective clusters. The outcomes would eventually converge since there exists a finite number of vertices to cluster and the objective function for all of them is the same. The following pseudocode represents the algorithm:

input : $R = \{r_1, ..., r_n\}$ (vectors to be clustered), k = number of clusters
output: $C = \{c_1, ...c_k\}$ (cluster centroids), $m : R \to C$ (cluster membership)

randomly set C to initial value
for *each* $r_i \in R$ **do**
    $m(\mathbf{r_i}) = \arg\min_{l \in \{1,...,k\}} ||\mathbf{r_i} - \mathbf{c_l}||^2$
end
while *m has changed* **do**
    for *each* $l \in \{1, ..., k\}$ **do**
        recompute $c_l = \frac{1}{|C_l|} \sum_{j \in C_l} \mathbf{r_j}$ as the centroid of $\{\mathbf{r_i} | m(\mathbf{r_i}) = l\}$
    end
    for *each* $r_i \in R$ **do**
        $m(\mathbf{r_i}) = \arg\min_{l \in \{1,...,k\}} ||\mathbf{r_i} - \mathbf{c_l}||^2$
    end
end
return *C, m*

4.2.2 Weighted k-means

The simple k-means algorithm is more suitable for the relaxation of the ratio cut. Another version is the weighted k-means in which the center of each cluster are both weighted. The weighted k-means should provide more accurate results, especially for the relaxation of the Ncut. Note that the simple k-means and the weighted k-means use the same objective function. There are different ways of calculating the weights. The default is to use the generalized degrees of the countries as weights. Using these weights, the weighted k-means would most definitely create more accurate results by definition of the objective function. However, we would like to try using population weights. We hope to examine whether population weights also help to yield better results. We calculate the weight of each country by dividing its population by the sum of the populations of all the included countries. We denote the weights by $d_1, \ldots, d_n$. The following pseudocode represents the algorithm:

input : $R = \{r_1, ..., r_n\}$ (vectors to be clustered), k = number of clusters,
$D = \{d_1, ..., d_n\}$ (weights of the vectors)
output: $C = \{c_1, ..., c_k\}$ (cluster centroids), $m : R \to C$ (cluster membership)

randomly set C to initial value
for *each* $r_i \in R$ **do**
    $m(\mathbf{r_i}) = \underset{l \in \{1,...,k\}}{\arg\min} ||\mathbf{r_i} - \mathbf{c_l}||^2$
end
while *m has changed* **do**
    for *each* $l \in \{1, ..., k\}$ **do**
        recompute $c_l = \frac{1}{\sum_{j \in C_l} d_j} \sum_{j \in C_l} d_j \mathbf{r_j}$ as the centroid of $\{\mathbf{r_i} | m(\mathbf{r_i}) = l\}$
    end
    for *each* $r_i \in R$ **do**
        $m(\mathbf{r_i}) = \underset{l \in \{1,...,k\}}{\arg\min} ||\mathbf{r_i} - \mathbf{c_l}||^2$
    end
end
return C, m

5 Clustering via the SVD for Directed Graphs

5.1 Motivation

Section 4 has conveyed how spectral decomposition of symmetric matrices is used to cluster vertices in an undirected graph. This method has been applied to cluster countries by their emmigration traits and immigration traits separately. Now we are interested in clustering vertices in a directed graph. Using the same migration dataset, if each edge in the graph represents the amount of migration from one country to another, then the edges would be directed. In this case, the directed graph contains information on both emmigration and immigration traits of the countries. See Figure 5 for the directed graph obtained by Mathematica using the modified dataset from Sheet 1 of the workbook "Inflows-nb".

One method for clustering directed graphs calls for forming a contingency table. This table is in matrix form $\mathbf{W_{n\times n}}$, and replaces the weight matrix in normalized spectral clustering. $\mathbf{W_{n\times n}}$ is defined by the following: for a fixed pair of vertices i and j where $i < j$,

- w_{ij} denotes some movement from i to j.
- w_{ji} denotes some movement from j to i.
- The diagonal entries of $\mathbf{W}$ are all zeros.

Note that this matrix can be rectangular after deleting the zero rows and columns.

Once we have a normalized form of $\mathbf{W}$, we can use its singular value decomposition to form clusters.

5.2 SVD of Matrices

In order to generalize the normalized spectral clustering methods for asymmetric, rectangular matrices, we introduce the singular value decomposition of a matrix $\mathbf{A}$. Suppose $\mathbf{A}$ is an $m \times n$ matrix with real-valued entries. We define the singular value decomposition of $\mathbf{A}$ as,

$$\mathbf{A} = \mathbf{V\Sigma U^T}$$

where

- $\mathbf{V}$ is an $m \times m$ orthogonal matrix with the left singular vectors of $\mathbf{A}$ as its columns
- $\mathbf{\Sigma}$ is an $m \times n$ matrix containing the singular values of $\mathbf{A}$ along its main diagonal and zeros otherwise. The number of singular values is generally the smaller value btween m and n.
- $\mathbf{U}^T$ is the transpose of an $n \times n$ orthogonal matrix $\mathbf{U}$, the matrix of the right singular vectors

If a matrix is $n \times n$ symmetric, then its singular value decomposition is essentially the same as the spectral decomposition. If the spectral decomposition of $\mathbf{A}$ is $\mathbf{A} = \mathbf{U\Lambda U^T}$, then the SVD is defined as the following:

- If $\lambda_i \geq 0$, $\sigma_i = \lambda_i$ and $\mathbf{u_i} = \mathbf{v_i}$
- If $\lambda_i < 0$, $\sigma_i = -\lambda_i$ and $\mathbf{u_i} = -\mathbf{v_i}$

If the ith eignevalue of $\mathbf{A}$ is positive or zero, than it is equal to the ith singular value of $\mathbf{A}$. In this case, the ith left and right singular vectors are equal to the ith eigenvector. On the other hand, if the ith eignevalue of $\mathbf{A}$ is negative, than its absoluate value is equal to the ith singular value of $\mathbf{A}$. In this case, the ith left and right singular vectors are opposite and any of the two can be the ith eigenvector.

5.3 The SVD of the Migration Dataset

The transformation of the contingency table and the SVD of the migration dataset follows the process below:

1. Form a square $q \times q$ matrix named $\mathbf{W}$ where q is the total number of countries in the migration network. The entries of $\mathbf{W}$ are defined in the following: for a fixed pair of countries i and j where $i < j$,
   - w_{ij} is the amount of people migrating from i to j.
   - w_{ji} is the amount of people migrating from j to i.
   - The diagonal entries of $\mathbf{W}$ are all zero.
2. Delete rows and columns whose entries are all zeros. The resulting $\mathbf{W}$ may be rectangular, with dimensions $m \times n$.
3. Form $\mathbf{D_{out}} = diag(d_{out,1}, \ldots, d_{out,m})$ where $d_{out,i} = \sum_{j=1}^{n} w_{ij}$, the total number of emigrants out of country i.
4. Form $\mathbf{D_{in}} = diag(d_{in,1}, \ldots, d_{in,n})$ where $d_{in,i} = \sum_{i=1}^{n} w_{ij}$, the total number of immigrants into country i.
5. Form the normalized contingency table $\mathbf{A} = \mathbf{D_{out}^{-\frac{1}{2}} W D_{in}^{-\frac{1}{2}}}$
6. Obtain the SVD of $\mathbf{A}$ as instructed in the previous section: $\mathbf{A} = \mathbf{V \Sigma U^T}$

5.4 Representation

Assuming $m > n$, take the diagonal entries of the singular value matrix $\mathbf{\Sigma}$ in the form: $1 = \sigma_0 > \sigma_1 \geq \cdots \geq \sigma_{n-1} \geq 0$ and find a sufficiently large gap between $\sigma_{k-1} > \sigma_k$. Again, this sufficiently large gap between $\sigma_{k-1} > \sigma_k$ is defined by the insignificance of all the gaps that follow it so it should be that last noticible difference. The number of desired bi-clusters is denoted by k.

We will cluster these countries twice, producing two different sets of k clusters. One set of k clusters has all the countries with emigrants (the number of countries $= m$). Another set of k clusters has all the countries with immigrants (the number of countries $= n$)

Form the matrices $\mathbf{\tilde{V}_{m \times k}} = (\mathbf{D_{out}^{-\frac{1}{2}} v_0}, \ldots, \mathbf{D_{out}^{-\frac{1}{2}} v_{k-1}})$ and $\mathbf{\tilde{U}_{n \times k}} = (\mathbf{D_{in}^{-\frac{1}{2}} u_0}, \ldots, \mathbf{D_{in}^{-\frac{1}{2}} u_{k-1}})$. Note that $\mathbf{D_{out}^{-\frac{1}{2}} v_0} = 1$ and $\mathbf{D_{in}^{-\frac{1}{2}} u_0} = 1$; they do not give us any nontrivial information about the vertices. Therefore, we disregard the first column of $\mathbf{\tilde{V}_{m \times k}}$ and the first column of $\mathbf{\tilde{U}_{n \times k}}$. Now, let $\mathbf{\tilde{v}_1}, \ldots, \mathbf{\tilde{v}_m}$ be the row vectors of $\mathbf{\tilde{V}}$ without the first component, these are the vector representations of the countries with emigrants. Let $\mathbf{\tilde{u}_1}, \ldots, \mathbf{\tilde{u}_n}$ be the row vectors of $\mathbf{\tilde{U}}$ without the first component, these are the vector representations of the countries with immigrants. A country can have two representative vectors, one from $\mathbf{\tilde{V}}$ and one from $\mathbf{\tilde{U}}$.

For more than two clusters ($k > 2$),

- A 2-D representation plot can be formed with points $(\tilde{v}_{i1}, \tilde{v}_{i2})$ $\forall i = 1, \ldots, m$ and $(\tilde{u}_{j1}, \tilde{u}_{j2})$ $\forall j = 1, \ldots, n$.
- If point $(\tilde{v}_{a1}, \tilde{v}_{a2})$ and point $(\tilde{v}_{b1}, \tilde{v}_{b2})$ are close to each other, then countries a and b have similar emigration traits.
- If point $(\tilde{u}_{a1}, \tilde{u}_{a2})$ and point $(\tilde{u}_{b1}, \tilde{u}_{b2})$ are close to each other, then countries a and b have similar immigration traits.
- If point $(\tilde{v}_{a1}, \tilde{v}_{a2})$ and point $(\tilde{u}_{b1}, \tilde{u}_{b2})$ are close to each other, then there exists a significant migration pattern from country a to country b.
- The 3-D representation plot can be formed with points $(\tilde{v}_{i1}, \tilde{v}_{i2}, \tilde{v}_{i3})$ $\forall i = 1, \ldots, m$ and $(\tilde{u}_{j1}, \tilde{u}_{j2}, \tilde{u}_{j3})$ $\forall j = 1, \ldots, n$

5.5 Clustering

Finally, we apply the k-means algorithm explained in section 3.3 to $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{U}}$ separately with the same number of clusters, k. As a result, a country with both an emigrant population and an immigrant population will be put into two different clusters: the emmigration trait cluster based on its emigrants, and the immigration trait cluster based on its immigrants. The clustering results should be shown in the same 2-D/3-D plots or in a matrix form where the columns and rows of the matrix are permuted separately according to their respective clusters.

6 Results

6.1 k-Means Clustering

6.1.1 Three Clusters

Figure 1 displays the 2-D representation plot of $k = 3$ discussed in section 5.5. Cluster 1, Cluster 2, and Cluster 3 of the emmigration trait clusters are represented by the e's, the f's, and the g's in the plot, respectively. And Cluster 1, Cluster 2, and Cluster 3 of the immigration trait clusters are represented by the 1's, the 2's, and the 3's in the plot, respectively. Without knowing which clusters the individual countries belong to, we can already see a significant correspondence between the e's and the 1's, between the f's and the 2's, and between the g's and the 3's. There clearly exists significant migration patterns between these specified clusters.

Now, let us look at the clustering results in the form of a matrix plot. The non-clustered matrix plot in Subfigure 2a displays the countries with immigrants as columns and the countries with emmigrants as rows. The matrix plot displays a darker color if the amount of migration between two countries is greater. Note that the countries are ordered quite randomly in the non-clustered matrix plot, mostly based on the country code system and how Mathematica inputs the data from the Excel workbook. On the other hand, the clustered matrix plot in Subfigure 2b permutes the columns and the rows so that countries that belong to the same cluster are put next to one another. The thin red lines divide

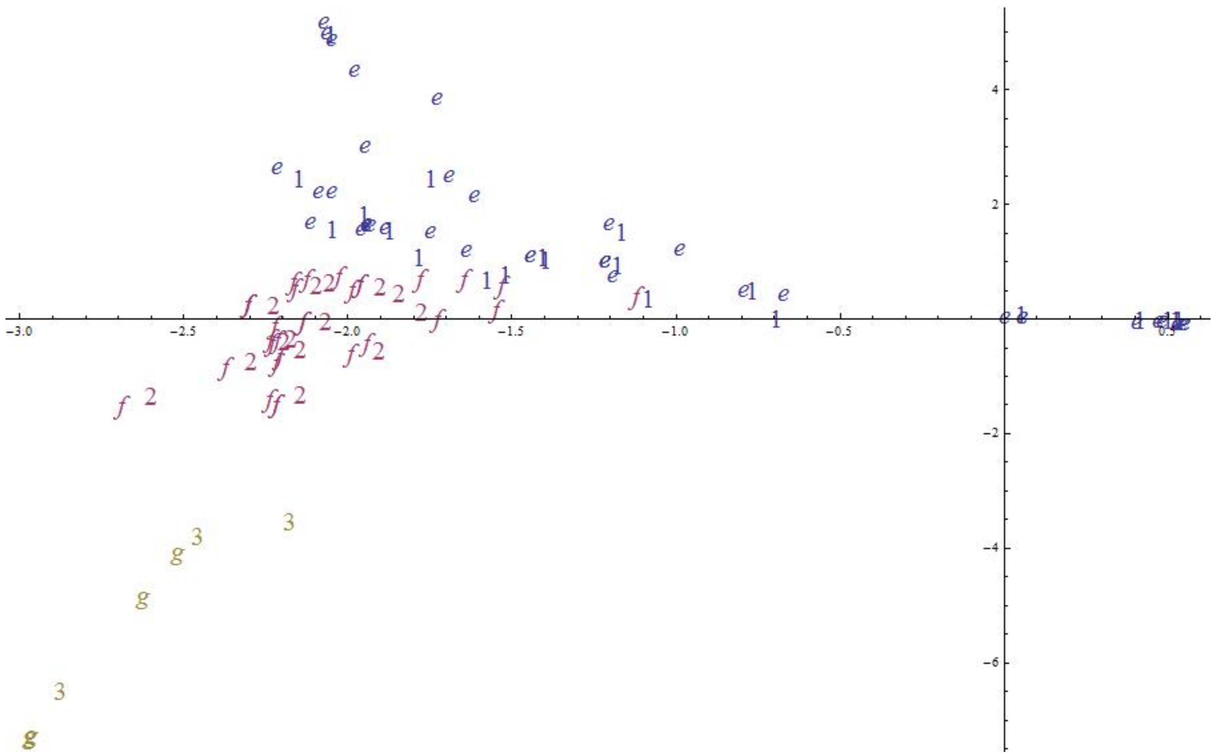
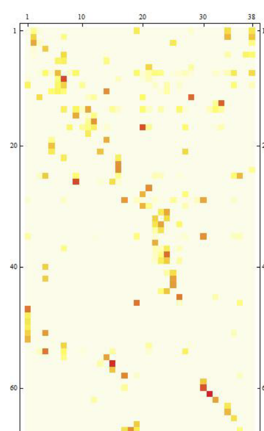
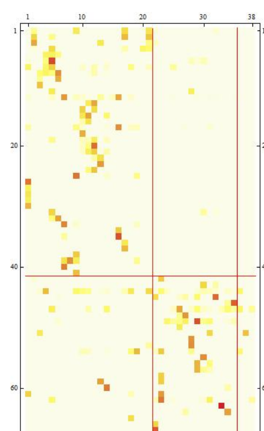


Figure 1: 2-D representation plot, $k = 3$



(a) Non-clustered matrix plot, $k = 3$



(b) Clustered matrix plot, $k = 3$

Figure 2: Matrix plots, $k = 3$

Cluster 1 (e's)	France, Portugal, Spain, Italy, Croatia, Poland, Turkey, Bosnia and Herzegovina, Macedonia, Slovenia, Greece, United States, Denmark, Norway, Sweden, Afghanistan, China, Iceland, Iran, Iraq, Somalia, Chile, Peru, Finland, India, Algeria, The Democratic Republic of Congo, Haiti, Lebanon, Tunisia, Hungary, Serbia, Japan, South Korea, Taiwan, The Netherlands, Belgium, New Zealand, Australia, South Africa
Cluster 2 (f's)	Argentina, Austria, Germany, Moldova, Belarus, Russia, Kazakhstan, Ukraine, Uzbekistan, Serbia and Montenegro, The Philippines, Sri Lanka, United Kingdom, Czech Republic, Slovakia, Vietnam, Ecuador, Colombia, Estonia, Morocco, Romania, Tajikistan, Lithuania, Suriname, Angola, Brazil, Cape Verde, Guinea Bissau, Canada, Armenia, Georgia
Cluster 3 (g's)	Azerbaijan, Bulgaria, Albania, Egypt

Table 1: Emmigration Trait Clusters, $k = 3$

Cluster 1 (1's)	France, Andorra, Croatia, Austria, Germany, Slovenia, The Philippines, Sri Lanka, United Kingdom, Denmark, Norway, Sweden, Ecuador, Finland, Ireland, Japan, Luxembourg, Netherlands, New Zealand, San Marino, Switzerland
Cluster 2 (2's)	Portugal, Spain, Poland, Moldova, Belarus, Russia, Cyprus, Czech Republic, Slovakia, Hungary, Romania, Kyrgyzstan, Latvia, Lithuania
Cluster 3 (3's)	Turkey, Macedonia, Greece

Table 2: Immigration Trait Clusters, $k = 3$

the clusters. As a result, we can see that our observation about the significant migration patterns between specified clusters from the 2-D representation plot is reconfirmed here.

Let us analyze the economic implications of the immigration trait clusters in Table 2. Countries are grouped together if they share similar migrant-sending countries. Cluster 1 has many countries from Western Europe and Nohtern Europe that have bigger economies and are more developed. Even though both Andorra and San Marino are extremely small countries, they both have very high GDP per capita. Hence their economic prosperity paired with the miniscule country size enables them to join the group of the more developed European countries. Cluster 1 only has two other European countries: Croatia and Slovenia, both from Central Europe. The rest of Cluster 1 are the odd cases: Zew Zealand, Japan, The Phillipines, Equador, and Sri Lanka. Japan's inclusion is easily justified since it is a developed country. However, it is hard to see more similarities between Japan and the Western/Northern European countries just by looking at the cluster itself. As for Zew Zealand, The Phillipines, Equador and Sri Lanka, their reasons for belonging in Cluster 1 is harder to see. For example, Equador's migrant-sending countries are mainly from South America so its ties with the rest of Cluster 1 countries is unclear. Insterestingly, New Zealand, The Phillipines, and Sri Lanka all have Australia as one of their migrant-sending countries, so we can see a reason why there countries are in the same cluster. Note that most of these odd-case countries have the United States as a migrant-sending country. Perhaps because of the larger amount of migration from the United States to these countries, the fact that the United States is shared between them is enough to declare that their immigration traits are similar. Overall, Cluster 1 demonstrates evidence for the dual labor market theory since most of the countries included are more developed, thus creating a strong pull factor for migrants.

Cluster 2 of the immigration clusters consists of mostly countries from Central and Eastern Europe. Latvia, Lithuania, and Russia are also included. There are two odd cases: Kyrgyzstan and Cyprus. They stand out because of their geographic locations. However, one possible explanation for their inclusion is the fact that Russia is a shared migrant-sending country among Kyrgyzstan and Cyprus as well as most of the countries in Cluster 2.

Cluster 3 of the immigration clusters has Turkey, Greece, and Macedonia. They are clustered together because of the extremely similar set of migrant-sending countries they have. In fact, these migrant-sending countries make up most of Cluster 3 of the emmigration trait clusters. See Table 1. As a result, there exists a nearly complete correspondence between Cluster 3 of the immigration trait clusters and Cluster 3 of the emmigration trait clusters.

The emmigration trait clusters in Table 1 are harder to analyze. There does not exist any clear explanation as to why such a large variety of countries are grouped together. Further economic research is necessary to apply the results to migration theories.

6.1.2 Four Clusters

The interpretation of the 2-D representation plot can be applied to the 3-D representation plot of $k = 4$ in Figure 3. Here, the emmigration trait clusters are represented by the small dots, each color represents a cluster. The immigration trait clusters are represented by the

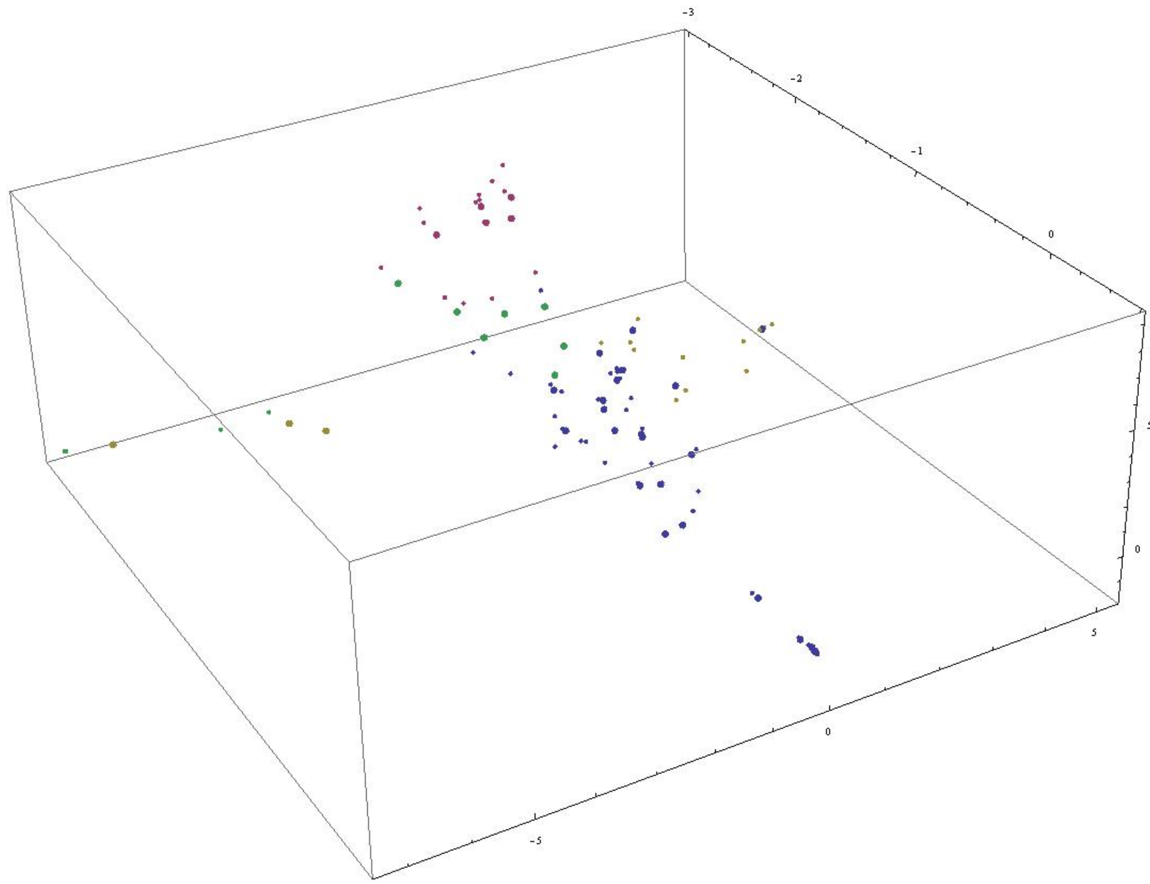


Figure 3: 3-D representation plot, $k = 4$

big dots. It is clear that the big dots and small dots of color blue are situated close to one another, showing significant migration patterns between the blue clusters. The same can be said about the purple clusters. However, it is more difficult to visualize the significant correspondence between the green clusters and between the yellow clusters.

Nevertheless, the clustered matrix plot in Figure 4 demonstrates that once the countries are permuted according to their respective clusters, significant migration patterns could be found between specified clusters.

The economic implications of the $k = 4$ clustering is unfortunately more difficult to interpret. Hopefully with further economic research, significant meaning can be obtained.

6.2 Weighted k-Means Clustering

Table 5 and Table 6 display the clustering results of $k = 3$ using the weighted k-means algorithm. Unfortunately, these clusters are not significant better then the clusters obtained with the simple k-means algorithm. In fact, they are even more difficult to interpret. For example, Cluster 2 of the emmigration trait clusters becomes overwhelmingly large compared to the other two clusters. This might be the result of data errors when obtaining the population weights of the countries. But more importantly, this shows that perhaps the

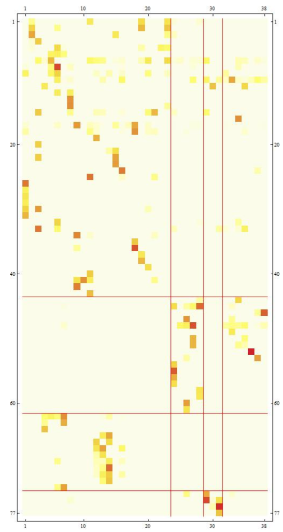


Figure 4: Clustered matrix plot, $k = 4$

Cluster 1	France, Portugal, Spain, Argentina, Italy, Croatia, Germany, Poland, Turkey, Russia, Serbia And Montenegro, Greece, Philippines, SriLanka, United Kingdom, Czech Republic, United States, China, Iceland, Ecuador, Chile, Colombia, Peru, Estonia, India, Algeria, Democratic Republic of Congo, Haiti, Lebanon, Morocco, Tunisia, Hungary, Romania, Japan, South Korea, Taiwan, Netherlands, Belgium, Suriname, New Zealand, Australia, North Korea, South Africa
Cluster 2	Austria, Moldova, Belarus, Kazakhstan, Ukraine, Uzbekistan, Slovakia, Vietnam, Tajikistan, Lithuania, Angola, Brazil, Cape Verde, Guinea Bissau, Canada, Israel, Armenia, Georgia
Cluster 3	Bosnia Herzegovina, Macedonia, Slovenia, Denmark, Norway, Sweden, Afghanistan, Iran, Iraq, Somalia, Finland, Serbia
Cluster 4	Azerbaijan, Bulgaria, Albania, Egypt

Table 3: Emmigration Trait Clusters, $k = 4$

Cluster 1	France, Andorra, Spain, Croatia, Austria, Germany, Slovenia, Cyprus, Philippines, Sri Lanka, United Kingdom, Denmark, Norway, Sweden, Ecuador, Finland, Ireland, Japan, Luxembourg, Netherlands, New Zealand, San Marino, Switzerland
Cluster 2	Portugal, Poland, Russia, Czech Republic, Romania
Cluster 3	Turkey, Macedonia, Greece
Cluster 4	Moldova, Belarus, Slovakia, Hungary, Kyrgyzstan, Latvia, Lithuania

Table 4: Immigration Trait Clusters, $k = 4$

Cluster 1	France, United States, China, India, Japan, South Korea, Taiwan, New Zealand, Australia, North Korea, South Africa
Cluster 2	Portugal, Spain, Argentina, Italy, Croatia, Austria, Germany, Poland, Turkey, Bosnia Herzegovina, Moldova, Belarus, Russia, Azerbaijan, Kazakhstan, Ukraine, Uzbekistan, Macedonia, Slovenia, Serbia And Montenegro, Greece, Philippines, SriLanka, United Kingdom, Bulgaria, Czech Republic, Slovakia, Vietnam, Denmark, Norway, Sweden, Afghanistan, Iceland, Iran, Iraq, Somalia, Ecuador, Chile, Colombia, Peru, Finland, Estonia, Algeria, Democratic Republic of Congo, Haiti, Lebanon, Morocco, Tunisia, Hungary, Romania, Serbia, Tajikistan, Lithuania, Netherlands, Belgium, Suriname, Angola, Brazil, Cape Verde, Guinea Bissau, Canada, Israel, Armenia, Georgia
Cluster 3	Albania, Egypt

Table 5: Weighted Emmigration Trait Clusters, $k = 3$

Cluster 1	France, Andorra, Portugal, Spain, Croatia, Austria, Germany, Poland, Moldova, Slovenia, Cyprus, Czech Republic, Slovakia, Denmark, Norway, Sweden, Ecuador, Finland, Hungary, Romania, Latvia, Lithuania, Luxembourg, Netherlands, San Marino, Switzerland
Cluster 2	Ukraine, Uzbekistan, Macedonia, United States, Vietnam, Iceland
Cluster 3	Poland, Bosnia Herzegovina, Moldova, Belarus, Azerbaijan, Denmark

Table 6: Weighted Immigration Trait Clusters, $k = 3$

generalized degrees of the countries are the more appropriate weights for this algorithm. Further investigation is necessary to improve the clustering results.

7 Summary

A Data

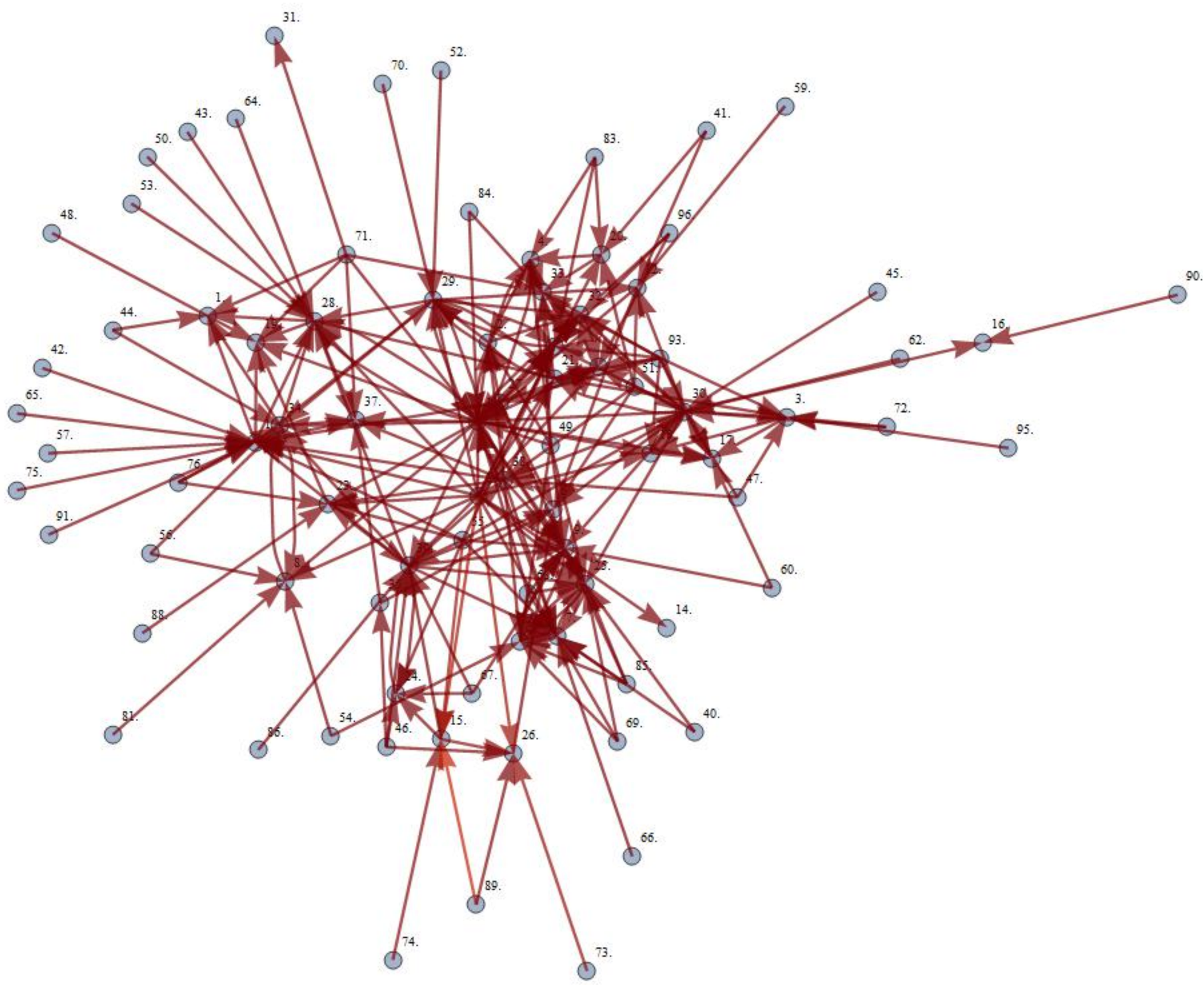


Figure 5: Directed graph of the migration dataset

Country of Destination	Country Code	Country of Origin	Country Code	Number of Migrants	Migration Weights	New Migratoin Weights
Andorra	1	[France]	10	579	9.78774E-05	9.78643E-05
Andorra	1	[Portugal]	28	2179	0.000368351	0.000368301
Andorra	1	[Spain]	34	2046	0.000345867	0.000345821
Andorra	1	[Argentina]	44	130	2.19759E-05	0.000021973
Andorra	1	[Italy]	71	122	2.06236E-05	2.06208E-05
Austria	2	[Croatia]	4	2535	0.000428531	0.000428473
Austria	2	[Germany]	11	16223	0.002742428	0.00274206
Austria	2	[Poland]	27	6035	0.001020191	0.001020054
Austria	2	[Turkey]	38	4897	0.000827817	0.000827706
Austria	2	[Bosnia and Herzegovina]	49	3235	0.000546863	0.000546789
Belarus	3	[Moldova, Republic of]	21	238	4.02329E-05	4.02275E-05
Belarus	3	[Russian Federation]	30	8091	0.001367749	0.001367565
Belarus	3	[Azerbaijan]	47	160	2.70473E-05	2.70437E-05
Belarus	3	[Kazakhstan]	72	234	3.95567E-05	3.95514E-05
Belarus	3	[Ukraine]	93	1435	0.000242581	0.000242548
Belarus	3	[Uzbekistan]	95	255	4.31066E-05	4.31009E-05
Croatia	4	[Germany]	11	107	1.80879E-05	1.80855E-05
Croatia	4	[Macedonia, The former Yugoslav Rep. of]	20	80	1.35237E-05	1.35218E-05
Croatia	4	[Slovenia]	33	66	1.1157E-05	1.11555E-05
Croatia	4	[Bosnia and Herzegovina]	49	387	6.54207E-05	6.54119E-05
Croatia	4	[Serbia and Montenegro]	83	182	3.07663E-05	3.07622E-05
Cyprus	5	[Greece]	12	1236	0.00020894	0.000208912
Cyprus	5	[Philippines]	26	1443	0.000243933	0.0002439
Cyprus	5	[Poland]	27	941	0.000159072	0.000159051
Cyprus	5	[Russian Federation]	30	290	4.90232E-05	4.90167E-05
Cyprus	5	[Sri Lanka]	35	1838	0.000310706	0.000310664
Cyprus	5	[United Kingdom]	39	1575	0.000266247	0.000266211
Cyprus	5	[Bulgaria]	51	282	4.76709E-05	4.76645E-05
Czech Republic	6	[Germany]	11	797	0.000134729	0.000134711
Czech Republic	6	[Moldova, Republic of]	21	2377	0.000401822	0.000401768
Czech Republic	6	[Poland]	27	949	0.000160424	0.000160403
Czech Republic	6	[Russian Federation]	30	4675	0.000790289	0.000790182
Czech Republic	6	[Slovakia]	32	6781	0.001146299	0.001146145
Czech Republic	6	[Ukraine]	93	30150	0.005096727	0.005096043
Czech Republic	6	[United States]	94	1804	0.000304958	0.000304917
Czech Republic	6	[Viet Nam]	96	6433	0.001087471	0.001087325
Denmark	7	[Germany]	11	2743	0.000463692	0.00046363
Denmark	7	[Norway]	25	1880	0.000317806	0.000317763
Denmark	7	[Sweden]	36	1589	0.000268614	0.000268578
Denmark	7	[Turkey]	38	506	8.55371E-05	8.55256E-05
Denmark	7	[United Kingdom]	39	1064	0.000179865	0.00017984
Denmark	7	[Afghanistan]	40	138	2.33283E-05	2.33252E-05
Denmark	7	[China]	55	1171	0.000197952	0.000197926
Denmark	7	[Iceland]	66	1584	0.000267768	0.000267732
Denmark	7	[Iran, Islamic Rep. of]	68	295	4.98685E-05	4.98618E-05
Denmark	7	[Iraq]	69	306	5.1728E-05	0.000051721
Denmark	7	[Somalia]	85	140	2.36664E-05	2.36632E-05
Denmark	7	[United States]	94	1840	0.000311044	0.000311002
Ecuador	8	[Germany]	11	548	9.2637E-05	9.26246E-05
Ecuador	8	[Spain]	34	2856	0.000482794	0.00048273
Ecuador	8	[Chile]	54	2606	0.000440533	0.000440474
Ecuador	8	[Colombia]	56	20841	0.003523081	0.003522608
Ecuador	8	[Peru]	81	8654	0.001462921	0.001462725
Ecuador	8	[United States]	94	15672	0.002649284	0.002648928
Finland	9	[Germany]	11	353	5.96731E-05	5.96651E-05
Finland	9	[Poland]	27	221	3.73591E-05	3.73541E-05
Finland	9	[Russian Federation]	30	2146	0.000362772	0.000362723
Finland	9	[Sweden]	36	749	0.000126615	0.000126598
Finland	9	[Turkey]	38	358	6.05184E-05	6.05102E-05
Finland	9	[United Kingdom]	39	285	4.8178E-05	4.81716E-05
Finland	9	[Bosnia and Herzegovina]	49	74	1.25094E-05	1.25077E-05
Finland	9	[China]	55	512	8.65514E-05	8.65398E-05
Finland	9	[Estonia]	60	2468	0.000417205	0.000417149
Finland	9	[India]	67	504	8.5199E-05	8.51876E-05
Finland	9	[Iran, Islamic Rep. of]	68	221	3.73591E-05	3.73541E-05
Finland	9	[Somalia]	85	287	4.85161E-05	4.85096E-05
Finland	9	[United States]	94	273	4.61495E-05	4.61433E-05
France	10	[Poland]	27	1119	0.000189162	0.000189137
France	10	[Turkey]	38	8760	0.00148084	0.001480641
France	10	[Algeria]	42	28454	0.004810026	0.00480938
France	10	[China]	55	11232	0.001898721	0.001898466
France	10	[Congo, Democratic Republic of]	57	1868	0.000315777	0.000315735
France	10	[Haiti]	65	3036	0.000513223	0.000513154
France	10	[Lebanon]	75	2254	0.000381029	0.000380978

France	10	[Morocco]	76	24054	0.004066225	0.004065679
France	10	[Tunisia]	91	10345	0.001748778	0.001748543
France	10	[United States]	94	4379	0.000740251	0.000740152
Germany	11	[Croatia]	4	8624	0.00145785	0.001457654
Germany	11	[Greece]	12	8624	0.00145785	0.001457654
Germany	11	[Hungary]	13	18654	0.003153378	0.003152955
Germany	11	[Poland]	27	152733	0.025818853	0.025815388
Germany	11	[Portugal]	28	5001	0.000845397	0.000845284
Germany	11	[Romania]	29	23743	0.004013651	0.004013113
Germany	11	[Russian Federation]	30	17081	0.002887469	0.002887082
Germany	11	[Turkey]	38	30720	0.005193083	0.005192386
Germany	11	[Bosnia and Herzegovina]	49	6635	0.001121618	0.001121468
Germany	11	[Iran, Islamic Rep. of]	68	3050	0.000515589	0.00051552
Germany	11	[Italy]	71	18293	0.003092353	0.003091938
Germany	11	[Serbia]	84	3745	0.000633076	0.000632991
Germany	11	[United States]	94	15435	0.00260922	0.00260887
Greece	12	[Romania]	29	5034	0.000850976	0.000850862
Greece	12	[Russian Federation]	30	2967	0.000501559	0.000501491
Greece	12	[Albania]	41	36841	0.006227812	0.006226976
Greece	12	[Bulgaria]	51	13210	0.002233093	0.002232794
Greece	12	[Egypt]	59	4843	0.000818688	0.000818578
Hungary	13	[Germany]	11	1176	0.000198798	0.000198771
Hungary	13	[Poland]	27	91	1.53832E-05	1.53811E-05
Hungary	13	[Romania]	29	6813	0.001151708	0.001151554
Hungary	13	[Russian Federation]	30	283	4.78399E-05	4.78335E-05
Hungary	13	[Slovakia]	32	930	0.000157212	0.000157191
Hungary	13	[China]	55	1466	0.000247821	0.000247788
Hungary	13	[Serbia and Montenegro]	83	1120	0.000189331	0.000189306
Hungary	13	[Ukraine]	93	2365	0.000399793	0.000399739
Hungary	13	[United States]	94	343	5.79827E-05	5.79749E-05
Hungary	13	[Viet Nam]	96	399	6.74492E-05	6.74402E-05
Ireland	14	[United States]	94	1300	0.000219759	0.00021973
Japan	15	[China]	55	589066	0.099579059	0.099565695
Japan	15	[Korea, Republic of]	74	199459	0.033717681	0.033713156
Japan	15	[Taiwan, China]	89	1282641	0.216824913	0.216795814
Japan	15	[United States]	94	753461	0.127369323	0.12735223
Kyrgyzstan	16	[Russian Federation]	30	147	2.48497E-05	2.48464E-05
Kyrgyzstan	16	[Tajikistan]	90	266	4.49661E-05	4.49601E-05
Latvia	17	[Belarus]	3	60	1.01427E-05	1.01414E-05
Latvia	17	[Germany]	11	223	3.76972E-05	3.76921E-05
Latvia	17	[Lithuania]	18	289	4.88542E-05	4.88476E-05
Latvia	17	[Russian Federation]	30	803	0.000135744	0.000135725
Latvia	17	[Estonia]	60	80	1.35237E-05	1.35218E-05
Latvia	17	[Ukraine]	93	76	1.28475E-05	1.28457E-05
Lithuania	18	[Belarus]	3	647	0.000109373	0.000109358
Lithuania	18	[Germany]	11	84	1.41998E-05	1.41979E-05
Lithuania	18	[Russian Federation]	30	398	6.72802E-05	6.72711E-05
Lithuania	18	[Ukraine]	93	294	4.96994E-05	4.96928E-05
Lithuania	18	[United States]	94	141	2.38354E-05	2.38322E-05
Luxembourg	19	[France]	10	2510	0.000424305	0.000424248
Luxembourg	19	[Germany]	11	929	0.000157043	0.000157022
Luxembourg	19	[Netherlands]	23	250	4.22614E-05	4.22557E-05
Luxembourg	19	[Portugal]	28	3796	0.000641697	0.000641611
Luxembourg	19	[Belgium]	48	911	0.000154001	0.00015398
Luxembourg	19	[Italy]	71	619	0.000104639	0.000104625
Macedonia, The former Y	20	[Albania]	41	210	3.54996E-05	3.54948E-05
Macedonia, The former Y	20	[Bulgaria]	51	79	1.33546E-05	1.33528E-05
Macedonia, The former Y	20	[Serbia and Montenegro]	83	283	4.78399E-05	4.78335E-05
Macedonia, The former Y	20	[United States]	94	67	1.13261E-05	1.13245E-05
Moldova, Republic of	21	[Romania]	29	171	2.89068E-05	2.89029E-05
Moldova, Republic of	21	[Russian Federation]	30	182	3.07663E-05	3.07622E-05
Moldova, Republic of	21	[Turkey]	38	443	7.48872E-05	7.48772E-05
Moldova, Republic of	21	[Ukraine]	93	354	5.98422E-05	5.98341E-05
Moldova, Republic of	21	[United States]	94	112	1.89331E-05	1.89306E-05
Netherlands	23	[Germany]	11	7150	0.001208677	0.001208514
Netherlands	23	[Turkey]	38	2768	0.000467918	0.000467856
Netherlands	23	[United Kingdom]	39	3583	0.000605691	0.000605609
Netherlands	23	[China]	55	2908	0.000491585	0.000491519
Netherlands	23	[Morocco]	76	1713	0.000289575	0.000289536
Netherlands	23	[Suriname]	88	997	0.000168539	0.000168516
Netherlands	23	[United States]	94	3121	0.000527592	0.000527521
New Zealand	24	[Japan]	15	2839	0.000479921	0.000479856
New Zealand	24	[United Kingdom]	39	14817	0.00250475	0.002504414
New Zealand	24	[Australia]	46	4791	0.000809898	0.000809789
New Zealand	24	[China]	55	4370	0.00073873	0.00073863

New Zealand	24	[India]	67	3093	0.000522858	0.000522788
Norway	25	[Denmark]	7	1493	0.000252385	0.000252351
Norway	25	[Finland]	9	573	9.68632E-05	9.68502E-05
Norway	25	[Germany]	11	2281	0.000385593	0.000385541
Norway	25	[Russian Federation]	30	1075	0.000181724	0.0001817
Norway	25	[Sweden]	36	3358	0.000567655	0.000567579
Norway	25	[United Kingdom]	39	971	0.000164143	0.000164121
Norway	25	[Afghanistan]	40	598	0.000101089	0.000101076
Norway	25	[Bosnia and Herzegovina]	49	133	2.24831E-05	2.24801E-05
Norway	25	[Iran, Islamic Rep. of]	68	279	4.71637E-05	4.71574E-05
Norway	25	[Iraq]	69	925	0.000156367	0.000156346
Norway	25	[Somalia]	85	1199	0.000202686	0.000202659
Norway	25	[United States]	94	739	0.000124925	0.000124908
Philippines	26	[Japan]	15	371947	0.062876032	0.062867593
Philippines	26	[Australia]	46	98041	0.016573407	0.016571183
Philippines	26	[Korea, Dem. People's Rep. of]	73	411539	0.069568888	0.069559551
Philippines	26	[Taiwan, China]	89	79461	0.013432538	0.013430736
Philippines	26	[United States]	94	627177	0.106021559	0.106007331
Poland	27	[Germany]	11	142	2.40045E-05	2.40013E-05
Poland	27	[Ukraine]	93	609	0.000102949	0.000102935
Portugal	28	[France]	10	159	2.68783E-05	2.68747E-05
Portugal	28	[Germany]	11	287	4.85161E-05	4.85096E-05
Portugal	28	[Moldova, Republic of]	21	2646	0.000447295	0.000447235
Portugal	28	[Romania]	29	1610	0.000272164	0.000272127
Portugal	28	[Spain]	34	249	4.20924E-05	4.20867E-05
Portugal	28	[United Kingdom]	39	827	0.000139801	0.000139782
Portugal	28	[Angola]	43	855	0.000144534	0.000144515
Portugal	28	[Brazil]	50	6036	0.00102036	0.001020223
Portugal	28	[Cape Verde]	53	1723	0.000291266	0.000291227
Portugal	28	[Guinea-Bissau]	64	615	0.000103963	0.000103949
Portugal	28	[United States]	94	98	1.65665E-05	1.65643E-05
Romania	29	[Austria]	2	75	1.26784E-05	1.26767E-05
Romania	29	[France]	10	125	2.11307E-05	2.11279E-05
Romania	29	[Germany]	11	252	4.25995E-05	4.25938E-05
Romania	29	[Hungary]	13	103	1.74117E-05	1.74094E-05
Romania	29	[Moldova, Republic of]	21	4349	0.00073518	0.000735081
Romania	29	[Canada]	52	187	3.16115E-05	3.16073E-05
Romania	29	[Israel]	70	156	2.63711E-05	2.63676E-05
Romania	29	[United States]	94	292	4.93613E-05	4.93547E-05
Russian Federation	30	[Lithuania]	18	69	1.16642E-05	1.16626E-05
Russian Federation	30	[Moldova, Republic of]	21	369	6.23779E-05	6.23695E-05
Russian Federation	30	[Armenia]	45	939	0.000158734	0.000158713
Russian Federation	30	[Azerbaijan]	47	667	0.000112753	0.000112738
Russian Federation	30	[China]	55	417	7.0492E-05	7.04826E-05
Russian Federation	30	[Georgia]	62	206	3.48234E-05	3.48187E-05
Russian Federation	30	[Kazakhstan]	72	1268	0.00021435	0.000214321
Russian Federation	30	[Ukraine]	93	2083	0.000352122	0.000352075
San Marino	31	[Italy]	71	328	5.5447E-05	5.54395E-05
Slovakia	32	[Austria]	2	430	7.26896E-05	7.26799E-05
Slovakia	32	[Czech Republic]	6	1294	0.000218745	0.000218716
Slovakia	32	[Germany]	11	913	0.000154339	0.000154318
Slovakia	32	[Hungary]	13	533	9.01013E-05	9.00893E-05
Slovakia	32	[Poland]	27	1132	0.00019136	0.000191334
Slovakia	32	[Romania]	29	396	6.69421E-05	6.69331E-05
Slovakia	32	[Russian Federation]	30	342	5.78136E-05	5.78059E-05
Slovakia	32	[Ukraine]	93	1007	0.000170229	0.000170206
Slovakia	32	[Viet Nam]	96	466	7.87753E-05	7.87647E-05
Slovenia	33	[Croatia]	4	1146	0.000193726	0.0001937
Slovenia	33	[Macedonia, The former Yugoslav Rep. of]	20	2097	0.000354489	0.000354441
Slovenia	33	[Moldova, Republic of]	21	83	1.40308E-05	1.40289E-05
Slovenia	33	[Russian Federation]	30	63	1.06499E-05	1.06484E-05
Slovenia	33	[Bosnia and Herzegovina]	49	7871	0.001330559	0.00133038
Slovenia	33	[Italy]	71	150	2.53569E-05	2.53534E-05
Slovenia	33	[Serbia]	84	4447	0.000751746	0.000751645
Slovenia	33	[Ukraine]	93	357	6.03493E-05	6.03412E-05
Spain	34	[Ecuador]	8	21387	0.00361538	0.003614895
Spain	34	[Germany]	11	16901	0.002857041	0.002856658
Spain	34	[Romania]	29	131457	0.022222237	0.022219255
Spain	34	[United Kingdom]	39	42535	0.007190358	0.007189393
Spain	34	[Argentina]	44	24191	0.004089384	0.004088835
Spain	34	[Colombia]	56	35621	0.006021576	0.006020768
Spain	34	[Morocco]	76	78512	0.013272114	0.013270333
Sri Lanka	35	[Australia]	46	31205	0.00527507	0.005274362
Sweden	36	[Denmark]	7	5137	0.000868388	0.000868271
Sweden	36	[Finland]	9	2639	0.000446112	0.000446052

Sweden	36	[Norway]	25	2492	0.000421262	0.000421205
Sweden	36	[Turkey]	38	1562	0.000264049	0.000264014
Sweden	36	[Bosnia and Herzegovina]	49	1058	0.00017885	0.000178826
Sweden	36	[Chile]	54	442	7.47182E-05	7.47082E-05
Sweden	36	[Iran, Islamic Rep. of]	68	2008	0.000339444	0.000339398
Sweden	36	[Iraq]	69	10850	0.001834146	0.001833899
Sweden	36	[Somalia]	85	2974	0.000502742	0.000502674
Switzerland	37	[France]	10	3500	0.00059166	0.00059158
Switzerland	37	[Germany]	11	9745	0.00164735	0.001647129
Switzerland	37	[Portugal]	28	5221	0.000882587	0.000882469
Switzerland	37	[Spain]	34	853	0.000144196	0.000144177
Switzerland	37	[Sri Lanka]	35	400	6.76183E-05	6.76092E-05
Switzerland	37	[Turkey]	38	978	0.000165327	0.000165304
Switzerland	37	[Italy]	71	2247	0.000379846	0.000379795
Turkey	38	[Germany]	11	9795	0.001655802	0.00165558
Turkey	38	[Russian Federation]	30	7784	0.001315852	0.001315675
Turkey	38	[United Kingdom]	39	7849	0.00132684	0.001326661
Turkey	38	[Azerbaijan]	47	12278	0.002075543	0.002075264
Turkey	38	[Bulgaria]	51	51683	0.008736788	0.008735615
United Kingdom	39	[France]	10	8237	0.001392429	0.001392242
United Kingdom	39	[Germany]	11	12632	0.002135385	0.002135098
United Kingdom	39	[Japan]	15	4008	0.000677535	0.000677444
United Kingdom	39	[New Zealand]	24	12182	0.002059314	0.002059038
United Kingdom	39	[Australia]	46	26004	0.004395864	0.004395274
United Kingdom	39	[China]	55	25927	0.004382847	0.004382259
United Kingdom	39	[India]	67	56850	0.009610247	0.009608957
United Kingdom	39	[South Africa]	86	16213	0.002740738	0.00274037
United Kingdom	39	[United States]	94	16055	0.002714028	0.002713664

Country of Destination	Country Code	Population in thousands	Population Weights (d_i)
[France]	10	65436	0.072663212
Andorra	1	81.222	9.01927E-05
[Portugal]	28	10585.9	0.011755081
[Spain]	34	43834.794	0.048676217
[Croatia]	4	4227.3	0.004694193
[Austria]	2	8155.138	0.009055849
[Germany]	11	82369	0.091466412
[Poland]	27	38216	0.042436844
[Turkey]	38	73639	0.081772209
[Moldova, Republic of]	21	3559	0.003952081
[Belarus]	3	9714.461	0.010787394
[Russian Federation]	30	142487	0.158224266
[Macedonia, The former Yugoslav Rep. of]	20	1618.482	0.001797239
[Slovenia]	33	2005.937	0.002227487
[Greece]	12	10702.664	0.011884741
Cyprus	5	736.928	0.00081832
[Philippines]	26	85358	0.094785538
[Sri Lanka]	35	20869	0.023173919
[United Kingdom]	39	59743.652	0.066342161
[Czech Republic]	6	10265	0.011398739
[Slovakia]	32	5389.2	0.005984421
[Denmark]	7	5574	0.006189632
[Norway]	25	4952	0.005498934
[Sweden]	36	9453	0.010497056
[Ecuador]	8	8940.108	0.009927516
[Finland]	9	5277	0.005859829
[Hungary]	13	9971	0.011072267
[Romania]	29	21597.289	0.023982645
Ireland	14	4232.9	0.004700411
[Japan]	15	127610	0.141704146
Kyrgyzstan	16	5507	0.006115232
Latvia	17	2294.6	0.002548032
[Lithuania]	18	3403.3	0.003779184
Luxembourg	19	517	0.000574101
[Netherlands]	23	135.25	0.000150188
[New Zealand]	24	4142.1	0.004599583
San Marino	31	31	3.44239E-05
Switzerland	37	7907	0.008780305

Country of Origin	Country Code	Population in thousands	Population Weights (d_i)
[France]	10	65436	0.013487105
[Portugal]	28	10585.9	0.002181875
[Spain]	34	43834.794	0.009034851
[Argentina]	44	24007.368	0.004948192
[Italy]	71	58435.04	0.012044128
[Croatia]	4	4227.3	0.000871295
[Austria]	2	8155.138	0.001680867
[Germany]	11	82369	0.016977189
[Poland]	27	38216	0.007876753
[Turkey]	38	73639	0.015177837
[Bosnia and Herzegovina]	49	3372	0.000695008
[Moldova, Republic of]	21	3559	0.00073355
[Belarus]	3	9714.461	0.002002261
[Russian Federation]	30	142487	0.029368194
[Azerbaijan]	47	8532.7	0.001758687
[Kazakhstan]	72	16558	0.003412792
[Ukraine]	93	46465.691	0.009577108
[Uzbekistan]	95	26312.7	0.005423347
[Macedonia, The former Yugoslav Rep. of]	20	1618.482	0.000333588
[Slovenia]	33	2005.937	0.000413446
[Serbia and Montenegro]	83	632	0.000130262
[Greece]	12	10702.664	0.002205941
[Philippines]	26	85358	0.017593256
[Sri Lanka]	35	20869	0.004301339
[United Kingdom]	39	59743.652	0.012313847
[Bulgaria]	51	7706.2	0.001588336
[Czech Republic]	6	10265	0.002115733
[Slovakia]	32	5389.2	0.001110776
[United States]	94	311591	0.064222455
[Viet Nam]	96	87840	0.018104825
[Denmark]	7	5574	0.001148865
[Norway]	25	4952	0.001020664
[Sweden]	36	9453	0.001948371
[Afghanistan]	40	35320	0.007279854
[China]	55	1314480	0.270929306
[Iceland]	66	319	6.57495E-05
[Iran, Islamic Rep. of]	68	74798	0.01541672
[Iraq]	69	32961	0.006793638
[Somalia]	85	9556	0.0019696
[Ecuador]	8	8940.108	0.001842658
[Chile]	54	17269	0.003559338
[Colombia]	56	45366.99	0.009350654
[Peru]	81	8339.199	0.001718804
[Finland]	9	5277	0.00108765
[Estonia]	60	1344.684	0.000277155

[India]	67	1241491	0.255885441
[Algeria]	42	35980	0.007415888
[Congo, Democratic Republic of]	57	4139	0.000853095
[Haiti]	65	10123	0.002086466
[Lebanon]	75	4259	0.000877828
[Morocco]	76	32272	0.006651627
[Tunisia]	91	10127.9	0.002087476
[Hungary]	13	9971	0.002055137
[Romania]	29	21597.289	0.004451447
[Serbia]	84	7261	0.001496575
[Albania]	41	3146.813	0.000648594
[Egypt]	59	72010.4	0.014842164
[Japan]	15	127610	0.026301875
[Korea, Republic of]	74	49779	0.010260019
[Taiwan, China]	89	23174	0.004776425
[Tajikistan]	90	7064	0.001455971
[Lithuania]	18	3403.3	0.000701459
[Netherlands]	23	135.25	2.78766E-05
[Belgium]	48	11008	0.002268874
[Suriname]	88	529	0.000109033
[New Zealand]	24	4142.1	0.000853734
[Australia]	46	20675.382	0.004261432
[Korea, Dem. People's Rep. of]	73	24451	0.00503963
[Angola]	43	19618	0.004043493
[Brazil]	50	156283.611	0.032211833
[Cape Verde]	53	500	0.000103056
[Guinea-Bissau]	64	1547	0.000318854
[Canada]	52	34482	0.007107133
[Israel]	70	7765	0.001600455
[Armenia]	45	3221.1	0.000663905
[Georgia]	62	9815	0.002022983
[South Africa]	86	50586	0.010426351

References

[1] "About Us." *The International Labour Organization*. N.p., n.d. Web. 06 Nov. 2012. <http://www.ilo.org/stat/Aboutus/lang–en/index.htm>.

[2] Bolla, M., Friedl, K., Krámli, A., "Singular value decomposition of large random matrices (for two-way classification of microarrays)," *Journal of Multivariate Analysis* **101** (2010), 434–446.

[3] Bolla, M., M.-Sáska, G,"Isoperimetric properties of weighted graphs related to the Laplacian spectrum and canonical correlations," *Studia Scientarum Mathematicarum Hungarica* **39** (2002), 425-441

[4] Bolla, M., "Spectra and structure of weighted graphs," *Electronic Notes in Discrete Mathematics* **38** (2011), 149–154.

[5] Bolla, M., Tusnády, G., "Spectra and optimal partitions of weighted graphs," *Discrete Mathematics* **128** (1994), 1–20.

[6] Butler, S., "Using discrepancy to control singular values for nonnegative matrices," *Lin. Alg. Appl.* **419** (2006), 486–493.

[7] "Demographic Yearbook." *United Nations Statistics Division*. N.p., 27 Mar. 2006. Web. 01 Oct. 2012 <http://unstats.un.org/unsd/demographic/products/dyb/dybcens.htm>.

[8] von Luxburg, U., "A Tutorial on Spectral Clustering," *Statistics and Computing* **17** (2007), 395–416.

[9] World Bank. "Population Data." *Google Public Data Explorer*. N.p., n.d. Web. 01 Oct. 2012. <http://www.google.com/publicdata/>.