Finding the Math Department's Deep Structure

Yuliy Baryshnikov, Haoyuan Li, Ning Jiang,
Anji Dong, Zifan Dong, Prajeet Basu,
Ziqi Xu, Adam Wawrowski, Qingyu Yi

December 20, 2023

Abstract

Spectral Analysis: Spectral team has collected four datasets on academic publications from the math faculty, including citations, journals, MSC codes, and references. Each dataset represents professors with their publication statistics. Our preprocessing involved removing columns with only a single non-zero entry, as they uniquely correspond to individual professors and don't contribute to clustering. We applied L1 and L2 normalization to mitigate data skewness, followed by PCA for dimensionality reduction. Our exploration of clustering methods, including K-means, Affinity Propagation, and Spectral Clustering, revealed that K-means on the MSC dataset provided the most coherent clusters. This suggests that MSC codes are effective in categorizing faculty research areas.

Barycenter Analysis: The Barycenter Team conducted a work that involves examining phylogenetic trees obtained from data supplied by a previous group in the UIUC Mathematics Department. We employ agglomerative clustering to produce phylogenetic trees. The goal is to determine the median or barycenter of these trees, which reflect references, MSC categories, and journals. The approach we use involves iterative operations utilizing the "pathtrees.py" script to accurately compute the barycenter. This study provides a comprehensive overview of the tree production process, including the mathematical techniques for determining the barycenter. Additionally, it explores potential avenues for future enhancements in our research.

1 Spectral Analysis

1.1 Dataset Introduction

In this project, we analyze four distinct datasets, each encompassing data related to 69 professors from the Mathematics Department. These datasets – Journals, References, Citations, and MSC – offer a comprehensive view of the professors' academic contributions and interactions within the field.

Journals Dataset: This dataset catalogs the publication records of the professors across various academic journals. Each column is named after a specific journal, such as "Annals of Combinatorics" or "Mathematische Annalen". The entries under each column signify

the number of times a professor (represented by each row) has published in that particular journal.

MSC Dataset: The MSC (Mathematics Subject Classification) dataset categorizes the professors' publications according to MSC codes. Each column represents a different MSC code. The entries denote the count of papers that a given professor (each row) has published under each MSC code.

Citations Dataset: This dataset focuses on the impact of the professors' work through citations. Columns in this dataset represent individual papers, identified by unique identifiers. The entries indicate the number of times a paper (in a specific column) has cited any work by the professor corresponding to a particular row.

References Dataset: Contrasting with the Citations dataset, the References dataset tracks the professors' engagement with existing literature. Like the Citations dataset, its columns represent papers by unique identifiers. However, the entries here reflect the number of times a professor (each row) has cited a specific paper (each column).

Together, these datasets provide a multifaceted view of the academic output and influence of the Mathematics Department's faculty, capturing both their contributions to and interactions with the broader mathematical community.

1.2 Data Preprocessing

Given the characteristics of our datasets, particularly in the Citations and References datasets, we observed a common occurrence: numerous columns, representing papers by their unique identifiers, contained only a single non-zero entry. This implies that these papers have limited interaction or influence in relation to the majority of the professors in our study. Such instances of minimal engagement do not contribute significantly to the analysis of academic interactions within the department.

To address this, we implemented a masking technique. This method involves excluding columns that have only one non-zero entry across all datasets. The rationale behind this approach is to focus our analysis on more impactful data, where the interactions or influences are more pronounced and widespread.

Then, we employed two different normalization techniques, L1 and L2 normalization, to create two distinct versions of each dataset.

L1 Normalization: Also known as least absolute deviations, this technique adjusts the values in a dataset so that the sum of the absolute values in each row equals one. It does this by dividing each value by the sum of the absolute values in its row. This approach is beneficial in our project as it helps in reducing the impact of outliers and ensuring that extreme values in the data do not disproportionately affect the overall analysis. L1 normalization enhances the robustness of our models against anomalies and offers a clear interpretation in terms of relative proportions of each professor's contribution or interaction.

L2 Normalization: Often referred to as least squares, L2 normalization works by dividing each value in a row by the square root of the sum of the squared values in that row. The result is that the sum of the squares of the values in each row is one. L2 normalization is particularly useful in our context as it preserves the geometric relationships in the data, such as the Euclidean distances between points (or professors, in our case). This is crucial

for applications where these relationships are meaningful, such as in clustering or principal component analysis (PCA).

By applying both L1 and L2 normalization to our datasets, we aim to leverage their distinct advantages. L1 normalization provides a robust model less sensitive to outliers, which is particularly beneficial when dealing with skewed data or anomalies. On the other hand, L2 normalization preserves data relationships, making it ideal for analyses involving distances or correlations. The use of both normalization techniques allows us to compare and contrast the outcomes and insights derived from each method, thereby enriching our understanding of the academic interactions within the Mathematics Department.

In the subsequent phase of our analysis, we employed Principal Component Analysis (PCA) to address the challenge of high dimensionality in our Citations and References datasets, each comprising over 30,000 columns. PCA is a sophisticated statistical technique designed to transform a complex dataset into a simpler structure without significantly losing the essence of the original data.

1.3 Introduction to Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical technique widely used in the fields of machine learning, data analysis, and statistics for the purpose of dimensionality reduction. By transforming large datasets into a simpler form, PCA not only enhances interpretability but also minimizes information loss. This method is particularly effective in simplifying the complexity inherent in high-dimensional data by distilling it into its most informative components, known as principal components.

1.3.1 How PCA Works

The essence of PCA lies in its ability to transform the original data into a new set of variables, the principal components, which are orthogonal to each other. These principal components are the directions in the data that maximize the variance, meaning they represent the most significant underlying structure in the data. In practical terms, PCA reorients the original data axes to these new axes, which are a better representation of the underlying patterns in the data.

1.3.2 Detailed Steps in PCA

1. **Standardization**: The process begins with the standardization of data. This step is crucial as it ensures that each feature contributes equally to the analysis, thereby neutralizing the effect of different scales of measurement.

2. **Covariance Matrix Computation**: After standardization, PCA computes the covariance matrix. This matrix is pivotal as it provides insights into how every pair of variables in the data are correlated. The covariance matrix thus forms the backbone of PCA, laying the groundwork for the extraction of principal components.

3. **Eigenvectors and Eigenvalues**: The core of PCA involves the calculation of eigenvectors and eigenvalues from the covariance matrix. Eigenvectors determine the directions of the new feature space, while eigenvalues dictate the magnitude

of variance along these new axes. The eigenvectors are orthogonal to each other, ensuring that the new feature space is fully uncorrelated.

4. **Data Projection**: The final step in PCA is to project the original data onto this new coordinate system formed by the eigenvectors. This projection effectively transforms the original high-dimensional data into a lower-dimensional form. The first few eigenvectors (principal components) are chosen as they correspond to the largest eigenvalues and thus represent the most significant variance in the dataset.

1.3.3 Deeper Intuition Behind PCA

The fundamental intuition behind PCA is its capability to uncover the underlying structure in a dataset. By reducing the number of variables, PCA not only simplifies the representation of the dataset but also preserves the most critical aspects of the data. This simplification is instrumental in exploratory data analysis, where it aids in visualizing complex data structures, and in predictive modeling, where it enhances model performance and interpretability. Additionally, PCA is often used to filter noise from the data, as the principal components tend to represent the signal, or the most informative part of the data, while leaving out the noise.

1.3.4 Applications and Advantages of PCA

PCA finds its applications in a multitude of domains, ranging from image processing and genomics to finance and marketing analytics. In image processing, for instance, PCA is used for feature extraction and image compression. In the field of genomics, PCA helps in identifying patterns in genetic variations. The versatility and efficacy of PCA make it an indispensable tool in the arsenal of data scientists and researchers.

1.4 Methods

After cleaning and preprocessing the datasets, we employed two distinct clustering techniques from the scikit-learn library: Affinity Propagation and K-Means. To determine the optimal number of clusters K in K-Means, we used the Elbow Method. This involves running the K-Means algorithm on the dataset for a range of values of K (e.g., from 1 to 30) and calculating the sum of squared distances from each point to its assigned center. When these overall intra-cluster distances are plotted against the number of clusters, the 'elbow' point where the rate of decrease sharply changes represents an appropriate number of clusters. The idea is that adding another cluster does not give much better modeling of the data after this point.

By applying these clustering techniques to our datasets, we aim to uncover meaningful patterns and groupings inherent in the data, facilitating a deeper understanding of the underlying structures within our academic datasets.

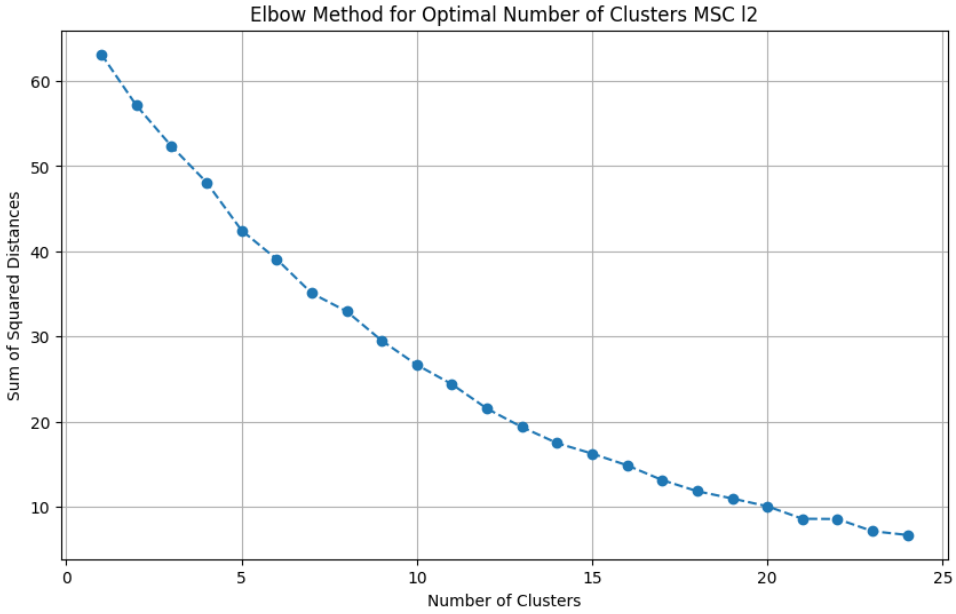

Figure 1: Elbow Plot Analysis

1.5 K-Means and Affinity Propagation Clustering Methods

1.5.1 K-Means Clustering

paragraphIntroduction and Mechanism K-Means is a widely-used clustering algorithm known for its simplicity and efficiency. The method partitions data into K distinct clusters based on feature similarities. It starts with the random initialization of K centroids, which are the proposed centers of the clusters. The algorithm then iteratively assigns each data point to the nearest centroid and recalculates the centroids based on the current cluster memberships.

Algorithmic Process

1. **Initialization**: K initial centroids are chosen randomly from the data points.
2. **Assignment**: Each data point is assigned to the nearest centroid, forming K clusters.
3. **Centroid Update**: Recompute the centroid of each cluster to be the mean of the points assigned to the cluster.
4. **Iteration**: Repeat the assignment and update steps until convergence, i.e., when the centroids no longer change significantly.

Applications and Challenges K-Means is employed in various domains like market segmentation, document clustering, and image compression. However, it has limitations, such as sensitivity to the initial centroid positions and difficulty in clustering data with non-spherical shapes or varying sizes and densities.

1.5.2 Affinity Propagation (AP)

Introduction and Mechanism Affinity Propagation (AP) is a clustering algorithm that identifies exemplars among data points and forms clusters based on these exemplars. Unlike K-Means, AP does not require the number of clusters to be specified in advance. It operates by sending messages between pairs of data points and iteratively refining the cluster assignments.

Algorithmic Process

1. **Responsibility and Availability Messages**: AP computes two types of messages – responsibility messages, which reflect the suitability of a data point being the exemplar for another, and availability messages, indicating how appropriate it is for a data point to choose another as its exemplar.
2. **Iterative Updating**: The algorithm iteratively updates these messages based on the data points' similarities and the current state of message values.
3. **Convergence**: The process continues until a high-quality set of exemplars and corresponding clusters emerges.

Applications and Limitations AP is particularly useful for complex datasets where the number of clusters is not known beforehand. It has been applied in fields like biology, computer vision, and information extraction. However, AP can be computationally intensive, especially for large datasets, and may produce a large number of small clusters.

1.5.3 Comparative Analysis and Use Cases

While both K-Means and AP are powerful clustering tools, their applications and suitability vary depending on the dataset characteristics. K-Means is often preferred for its simplicity and speed, especially in large datasets with well-defined, spherical clusters. In contrast, AP's ability to handle complex structures and automatically determine the number of clusters makes it suitable for datasets where such pre-specifications are challenging. The choice between K-Means and AP should be guided by the nature of the data and the specific requirements of the analysis.

The application of these clustering methods extends beyond academic research into practical industry solutions, where they are used to uncover patterns, segment markets, and inform decision-making processes. As with any algorithm, understanding their underlying assumptions and limitations is key to effectively leveraging their capabilities.

1.6 Results and Future Developments

Since most of our clusterings are all over 10 subgroups, it's hard to visualize all the results here. Below is the result clustering with l2 normalized MSC dataset with 30 PCA components number, and 15 K-means clustering number.

- **Cluster 0**: DeVilleRELee, RaptiZoi.

- **Cluster 1**: BradlowStevenBenjamin, HaboushWilliamJ, HellerJeremiahBen, JandaFelix, KatzSheldonH, YongAlexanderTF.
- **Cluster 2**: AlbinPierre, FernandesRuiLoja, HungPei-Ken, KermanEly, LermanEugeneM, NikolaevIgorG, PascaleffJamesThomas, TolmanSusan.
- **Cluster 3**: AhlgrenScottD, BocaFlorin-Petre, DuursmaIwanMaynard, FordKevinB, ReznickBruce, ThornerJesse, ZaharescuAlexandru.
- **Cluster 4**: DiFrancescoPhilippe, KirkpatrickKayLene, LeditzkyFelix, YoungAmanda.
- **Cluster 5**: DunfieldNathanM, GuzmanRosemaryK, RasmussenJacob, RasmussenSarahDean.
- **Cluster 6**: BaryshnikovYuliyM, DeyParthaSarathi, SongRenming, SowersRichardB, WeiWei, WuXuan.
- **Cluster 7**: CooneyDanielB, FadinaTolulope, FengRunhuan, JingXiaochen, QuanZhiyu.
- **Cluster 8**: BronskiJaredC, ErdoğanMehmetBurak, HurVeraMikyoung, KirrEduard, LaugesenRichardSnyder, LiXiaochun, TzirakisNikolaos, ZharnitskyVadim.
- **Cluster 9**: BaloghJózsef, KostochkaAlexandrV.
- **Cluster 10**: JungeMarius, KutzarovaDenkaN, OikhbergTimur.
- **Cluster 11**: IvanovSergeiVladimirovich, LiuYuan, MineyevIgor.
- **Cluster 12**: Berwick-EvansDaniel, HiraniAnilN, McCarthyRandy, RezkCharlesW, StojanoskaVesna.
- **Cluster 13**: HinkkanenAimo, TysonJeremyT.
- **Cluster 14**: DoddChristopher, DuttaSankarPrasad, KedemRinat, LaNaveGabriele, TumanovAlexander.

But we have uploaded all our clustering results as CSV files for further analysis. During our analysis process, the MSC dataset emerged as the most effective in categorizing faculty, while other datasets like journals or citations showed potential biases. So far, our limitation has been the inability to find an effective way to combine all the different academic publication statistics together. We are only able to analyze them separately. A future direction could involve improving the preprocessing of these datasets and exploring methods to integrate them effectively with varying weights.

2 Barycenter Analysis

2.1 Data Preprocessing

2.1.1 Similarity Matrices

We create the corresponding similarity matrices using the Journals dataset, References dataset, and MSC dataset gathered from the following sources and steps.

1. Download data from the MathSciNet website and save it in JSON format files using the following sub-steps:

   a. Save netid and user password to the `credential.py` file. Inspect ids for buttons on the login pages, use the `.click()` function and `.sleep()` from the webdriver package to perform auto sign-in.

   b. Inspect the id of the search bar, search for publications of math professors using the `.click()` and `.sleep()` functions.

   c. Click on each publication to download the information of references, MSC Code, and Journal for the corresponding paper.

   d. Save the information for each professor in a JSON file named `[professor]_papers.json` (e.g., `TolmanSusan_papers.json`) in the following format:

```
{

"PaperA": {"Title", "PaperID", "Author", "
    Journal_Name", "Publication_Year", "
    References", "Codes"},

"PaperB": {"Title", "PaperID", "Author", "
    Journal_Name", "Publication_Year", "
    References", "Codes"},
...
}
```

   where "Codes" refers to the MSC category code.

2. Extract raw data on MSC codes, References, and Journals for each professor in the department:

   a. Extract the "Journal_Name" index for Journals, "References" index for References, and "Codes" for MSC Codes from the `[professor]_papers.json` files.

Similarity Matrix for Journals For any two professors i and j, the similarity matrix S for journals is defined as:

$$S[i, j] = \text{Number of unique journals in common between prof } i \text{ and prof } j$$

For example, if professor A published papers in journals X and Y, while professor B published in journals X and Z, then the entry $S[A, B]$ of this matrix would be 1.

Similarity Matrix for MSC The MSC similarity matrix S considers both main and side codes used by the professors:

$$S[i, j] = 2 \cdot (\text{Number of main codes in common}) + 1 \cdot (\text{Number of side codes in common})$$

Note: Only the first two digits of each MSC code are considered.

For example, if professor A's codes are [X, Y, (Z), (W)] and professor B's are [Y, (Z), T, (U)], then the entry $S[A, B]$ would be $2 + 1 = 3$ as they share the main code Y and the side code (Z).

Similarity Matrix for References For references, the similarity matrix S is given by:

$$S[i, j] = \text{Number of times prof } i \text{ and prof } j \text{ have cited the same reference}$$

For instance, if professor A referred to papers x, y, z in her papers, and professor B referred to papers w, x, y in his papers, then the entry $S[A, B]$ of this matrix would be 2. The benefit of this method is regardless of how many time A sited x

2.1.2 Distance Matrices

The distance matrix is calculated using the following formula:

$$d(A, B) = \sqrt{1 - \left(\frac{\langle v_A, v_B \rangle}{\|v_A\| \|v_B\|} \right)}$$

The computation of the distance matrix begins by initializing a zero matrix that has the same dimensions as the input similarity matrix. For each pair of profiles, which correspond to the rows in the similarity matrix, the distance is calculated using the specified formula above. This calculation involves finding the inner product of the vectors representing the profiles and normalizing it by the product of their Frobenius norms. After computing these distances for all pairs of profiles, the diagonal elements of the matrix are set to zero. These elements represent the distance of a profile with itself, which is logically zero. Finally, the fully computed distance matrix, reflecting the distances between all pairs of profiles, is returned as the output.

Heatmap is a map representing the frequency of individual values as a color. Below is the heatmap for the distance matrices. In this case, red means a relatively close distance between the two professors. Blue means a relatively long distance between the two professors.

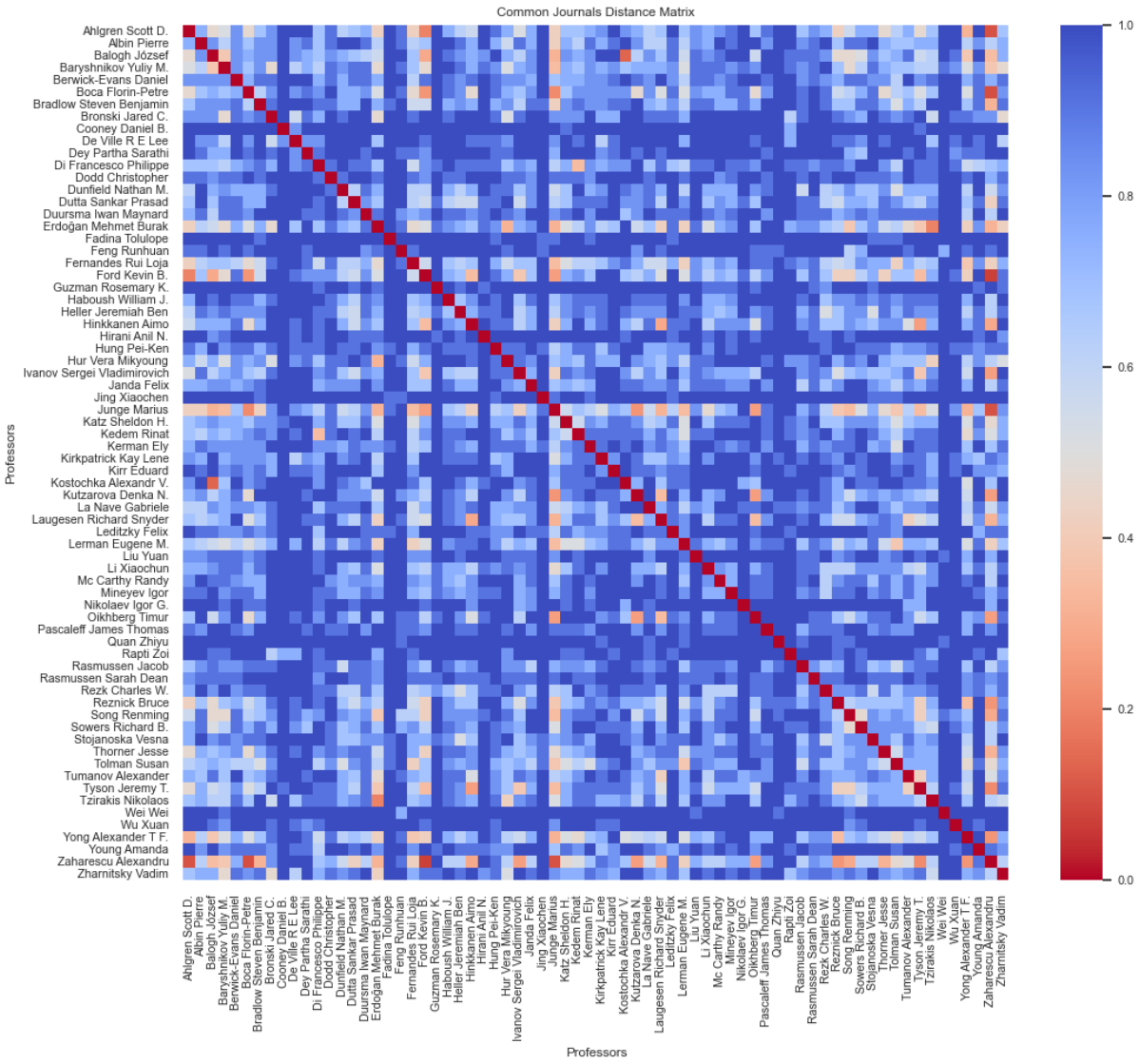

Figure 2: Heatmap for Journal Distance Matrix

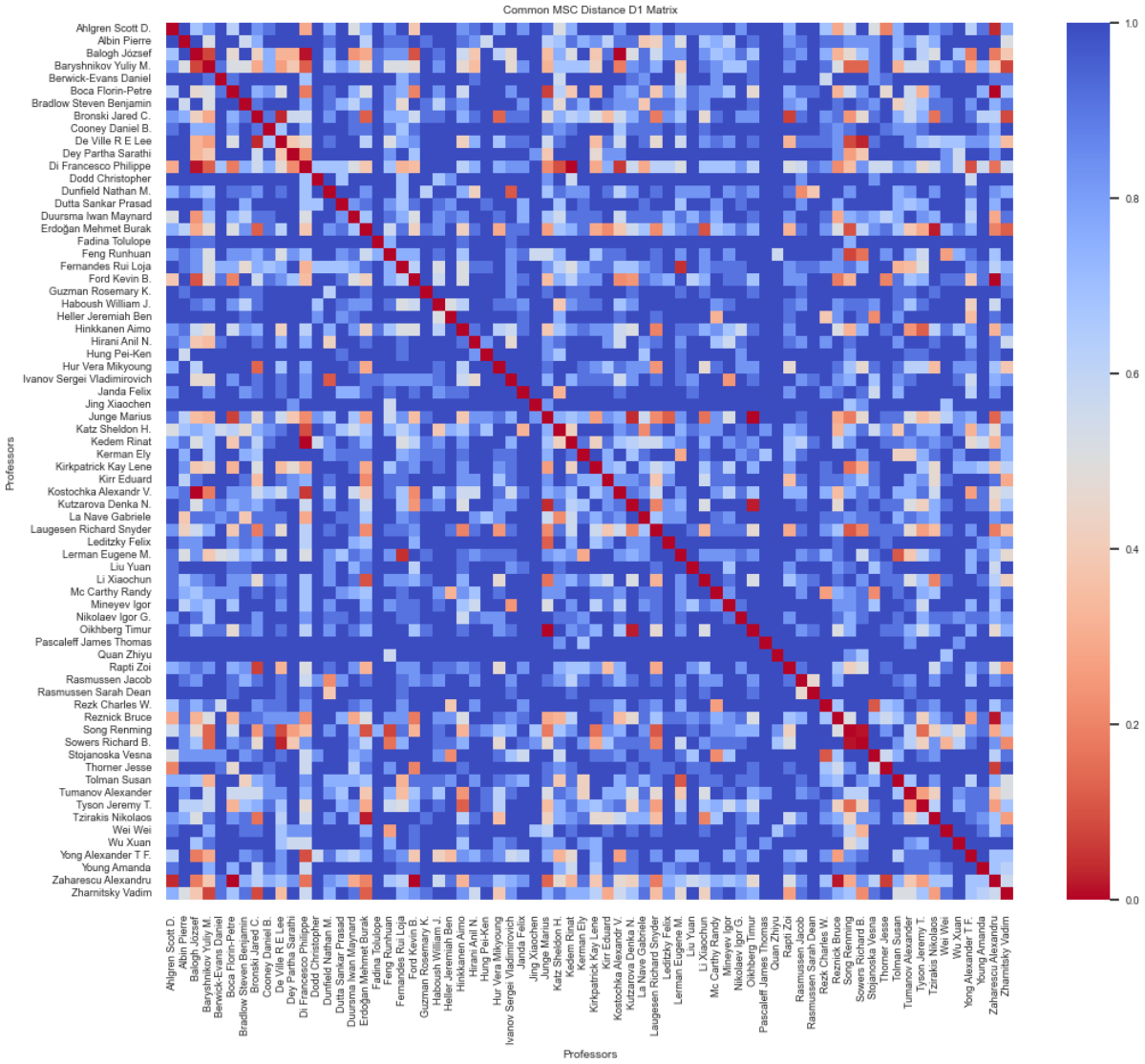

Figure 3: Heatmap for MSC Distance Matrix

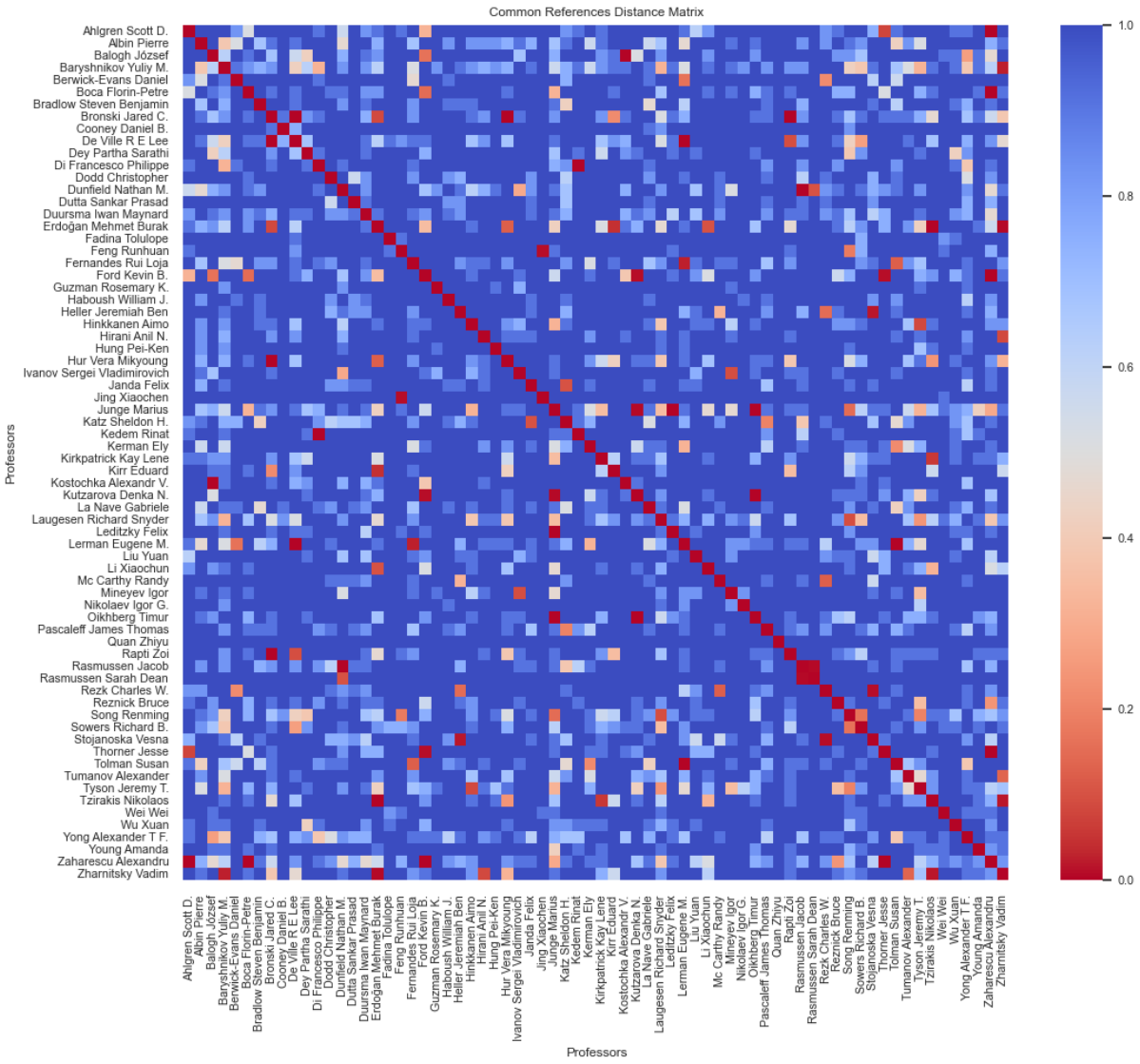

Figure 4: Heatmap for Reference Distance Matrix

2.2 Methodology

2.2.1 Phylogenetic tree and Newick Format tree

A phylogenetic tree is a diagrammatic representation that showcases the evolutionary relationships among various biological species or entities based upon similarities and differences in their physical or genetic characteristics. The structure of a phylogenetic tree is a branching system where each branch point, or node, represents the common ancestor of the species diverging from that point. The length of the branches can reflect the genetic changes or chronological time, depending on the type of phylogenetic tree constructed. These trees are constructed through the analysis of morphological or genetic data, where algorithms and statistical models determine the likelihood of evolutionary paths.

The Newick format is a way of representing graph-theoretical trees with a specific syntax in a text form, primarily used to denote phylogenetic trees. In the Newick format, trees are represented as a nested set of parentheses with branch lengths specified by numbers.

Each set of parentheses corresponds to a node, and the enclosed leaf nodes and subnodes represent descendants of that node. Leaves in the tree are typically represented by the name of the data item—like the species or genes they represent—while internal nodes may be named or left blank if the focus is on the tree's structure. For example, a simple tree with three descendants might be represented as ((A:0.1,B:0.2):0.3,C:0.4);, where A, B, and C are the leaf nodes, and the numbers represent branch lengths.

2.2.2 Billera-Holmes-Vogtmann (BHV) treespace and shortest paths

The Billera-Holmes-Vogtmann (BHV) treespace is a concept from the mathematical field of geometric phylogenetics, which integrates geometry with evolutionary biology. In mathematics, the BHV treespace is defined as a space where each point represents a phylogenetic tree, a tree that depicts the evolutionary relationships among a set of species or taxa.

The Geodesic Treepath Problem (GTP) algorithm is a polynomial algorithm to compute the geodesic distance between two phylogenetic trees, which was introduced by Billera et al. (2001).

In BHV treespace, there exists a unique shortest path, or geodesic, that joins two phylogenetic trees T1 and T2. This geodesic may be calculated using the Geodesic Treepath Problem (GTP) technique (Owen and Provan., 2011).

2.2.3 Iteration and Barycenter

The general process involves the following steps:

1. Compare the data between each professor to quantify their similarity, and then store the data into a similarity matrix or distance matrix.
2. Generate the initial phylogenetic trees for each distance matrix using hierarchical clustering, along with their corresponding heat maps.
3. Use the `pathtrees.py` file in the pathtrees repository to iteratively generate the barycenter of the trees:
    a. Initialize with `numpathtrees = 3` and the starting trees MSC and REF. The output is the first pathtree.
    b. Take the output from the previous step, increase `numpathtrees` to 4, and include the Journal tree to generate two pathtrees.
    c. For $N > 0$ iterations:
        i. Input `numpathtrees = 3 + currN` and the last output tree, alternating with the trees MSC, REF, and Journal depending on the iteration stage.
        ii. The specific tree can be found by taking `currStartTree = currN (mod 3)`, where we map `REF: 0, Journal: 1, MSC: 2`.
        iii. The final output after N iterations will be the barycenter of the trees.

    This iterative process can be repeated to find increasingly precise barycenters.

4. Aggregate the iterations into a single file to transform into visualized trees, with the final iteration representing the current barycenter, using tree plotting code.

The methodology starts with the generation of similarity and distance matrices from the provided data, followed by the creation of phylogenetic trees using agglomerative clustering techniques. The trees are then converted into Newick format for further analysis. We offer a method to build intermediate trees on the shortest path between two arbitrary trees, called pathtrees, based on the Billera-Holmes-Vogtmann (BHV) distance between pairs of trees. These pathtrees give a structured way to investigate intermediate neighboorhoods between trees of interest in the BHV tree space. The method uses the Java package GTP (Owen and Provan., 2011) to generate the geodesic between pairs of trees, pathtrees for generating intermediate trees on the shortest path between two arbitrary trees in the Pathtrees Github Repository (Khodaeil et al., 2023) and several other standard Python modules. Using iteration to find reasonable median, the barycenter, of three trees.

Algorithm 1 Generates the Barycenter between MSC, References, and Journals

Require: $N > 0$
  $MSC \leftarrow$ MSC Tree
  $REF \leftarrow$ Reference Tree
  $JOUR \leftarrow$ Journal Tree
  $numpathtrees \leftarrow 3$
  $Output0 \leftarrow$ `pathtrees`$(MSC,\ REF,\ numpathtrees)$
  $numpathtrees \leftarrow 4$
  $Output1 \leftarrow$ `pathtrees`$(OutPut0,\ JOUR,\ numpathtrees)$
for $i = 0;\ i < N;\ i \leftarrow i + 1$ **do**
  $numpathtrees \leftarrow 4 + i$
  $x \leftarrow i + 2$
  $Outputx \leftarrow$ `pathtrees`$(CurrStartTree, OutputN(previous), numpathtrees)$

2.3 Results and Future work

After completing 17 iterations, our algorithm has yielded an initial barycenter of the phylogenetic trees. This barycenter is a significant stride towards decoding the intricate academic focus areas within the department. Future enhancements may involve the integration of additional perspectives, coupled with an increased number of iterations, to achieve a closer convergence to the true barycenter. Such advancements could unveil deeper structural insights and connections within the UIUC Mathematics Department.

The 3 original phylogenetic trees for journal, MSC and reference.

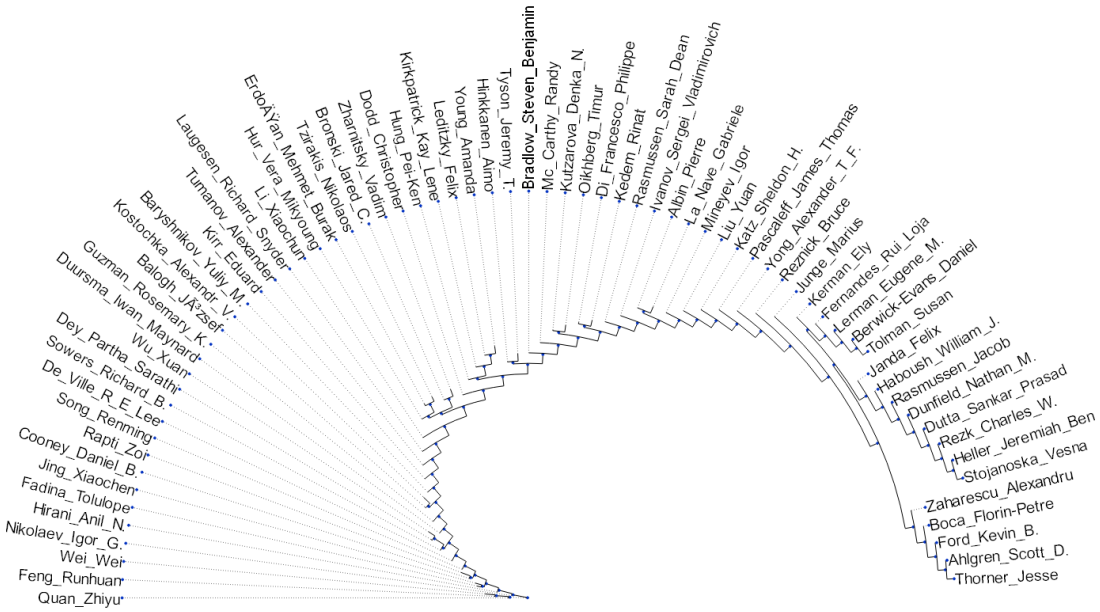

Figure 5: Journal phylogenetic trees

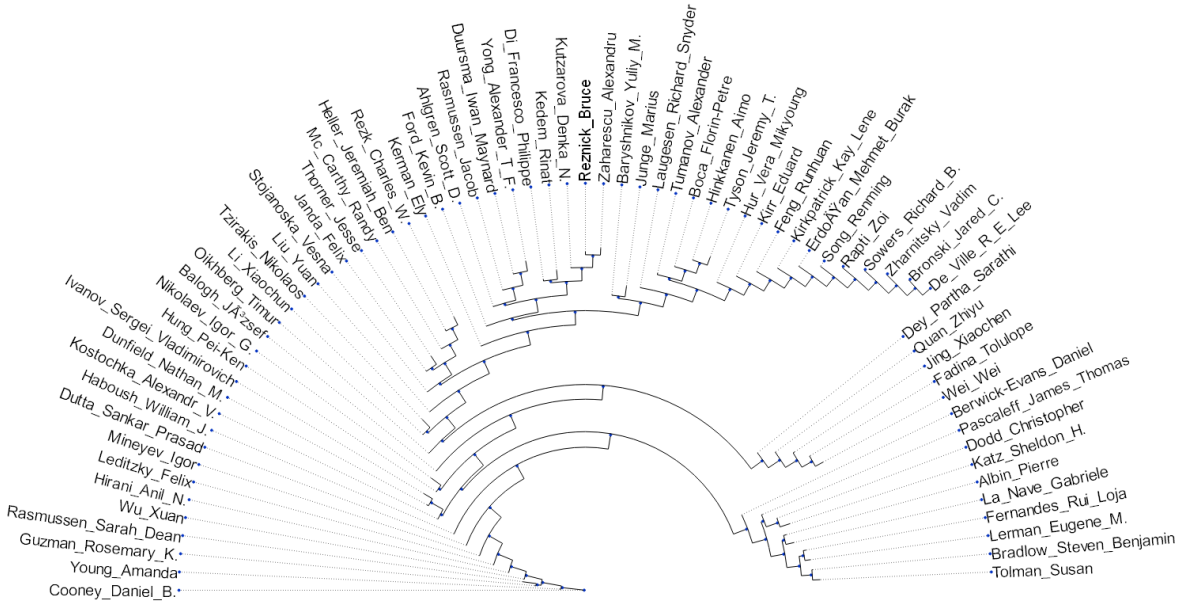

Figure 6: MSC phylogenetic trees

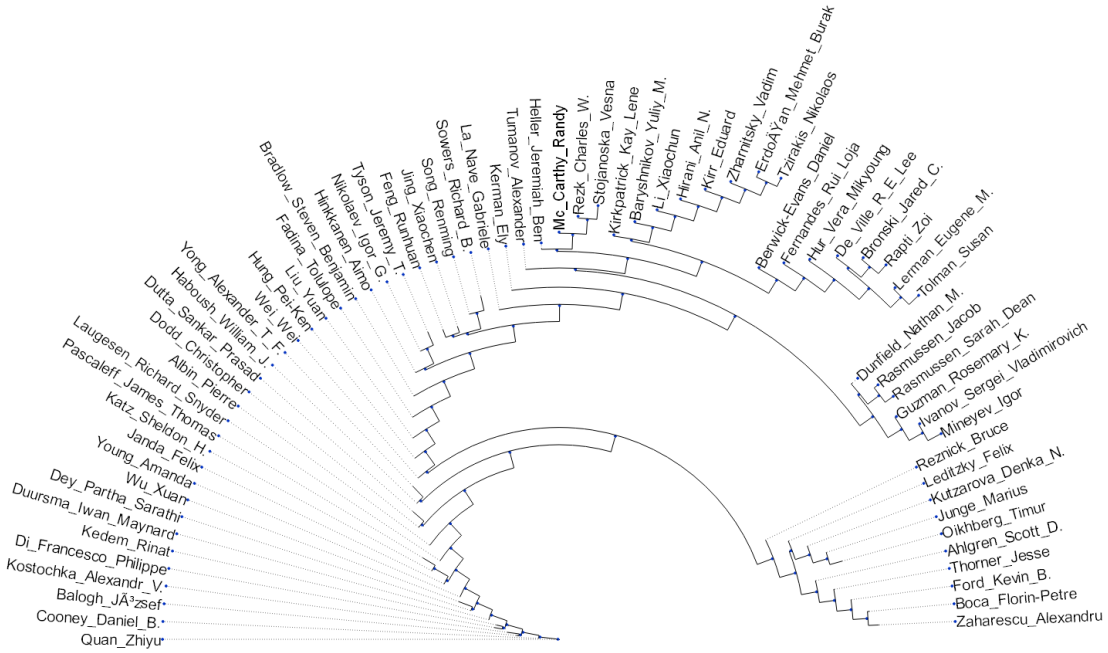

Figure 7: Reference phylogenetic trees

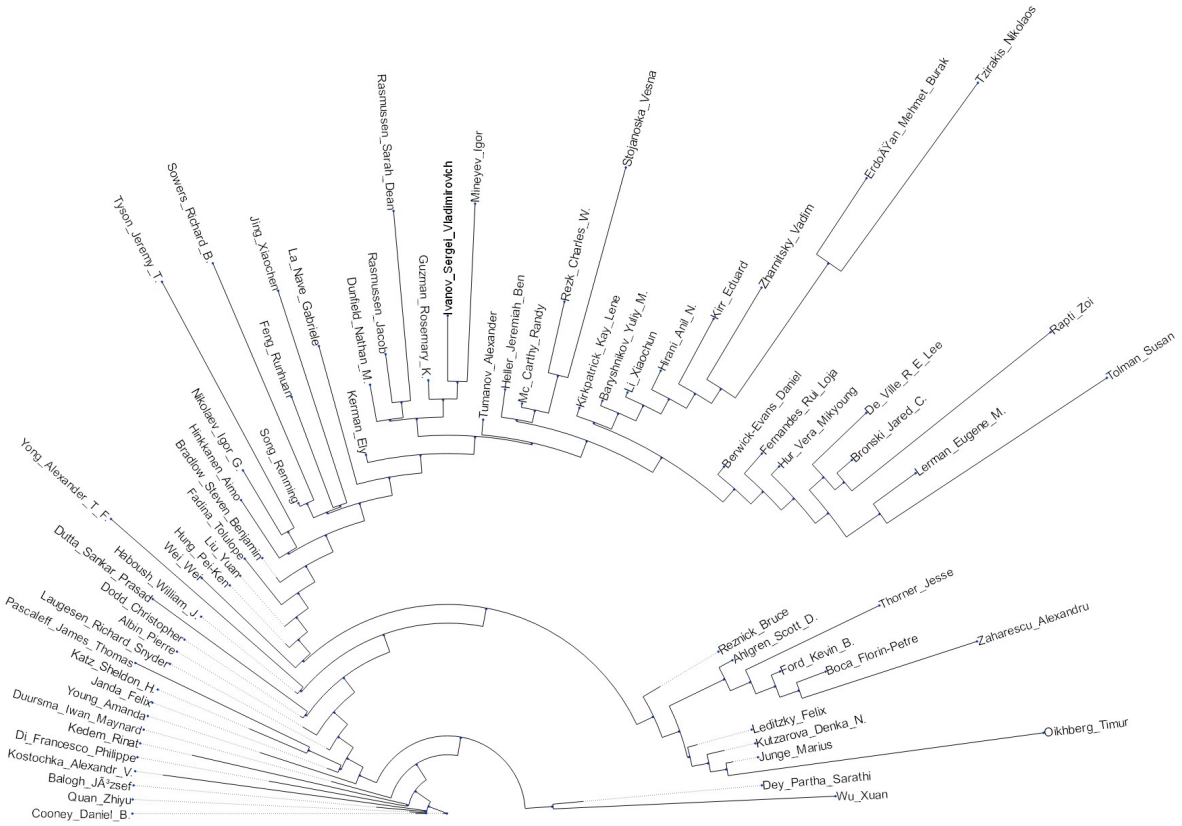

Figure 8: final barycenter after 17 iterations

References

Billera, L., Holmes, S., and Vogtmann., K. (2001). Geometry of the space of phylogenetic treesn. *Advances in Applied Mathematics*, 27:733–767.

Khodaeil, M., Owen, M., and Beerli, P. (2023). Pathtrees. `https://github.com/TaraKhodaei/PATHTREES`.

Owen, M. and Provan., J. S. (2011). A fast algorithm for computing geodesic distances in tree space. *IEEE/ACM Trans Comput Biol Bioinf*, 8(1):2–13.