Group 4: Victoria Hannett, Ruoxi Wang, Swas Tripathi, and Alan Gu
CIT 5900 - 002
May 9, 2025

Project 3 Report: Putting it All Together

1 Project Summary

The purpose of this project is to consolidate the findings of all groups' searches to create a large inventory of research outputs related to the Federal Statistical Research Data Centers (FSRDCs). This code is meant to create a streamlined list of research outputs and to investigate any relationships that exist within the data. Finally, this project involves the creation of an open-source Github Pages site to display the information.
GitHub landing page: https://rosiewang1224.github.io/CIT5900-Project3/

2 Input Processing

2.1 Why Process Inputs

Due to differing methodologies, the Project 2 outputs vary between each group. The web searches to identify research outputs centered on different factors, such as researchers that utilized FSRDC data, institutions that publish the most FSRDC data, and the common topics among FSRDC publications. In short, input processing served four purposes: 1) remove duplicate works across all the data from different groups, 2) enrich the data with information about each work via web scraping, 3) add information related to the overarching project under which each work was published, and 4) filter out projects that were not FSRDC-related.

2.2 Data Enrichment: enrich_all_csv_files.py

To perform the data enrichment, we used the titles from each CSV file to web scrape the APIs OpenAlex, CrossRef, and arXiv sequentially. We then used the metadata obtained from the web scraping to create the following columns:

- OutputTitle - title of the work
- Authors* - authors of the work
- OutputBiblio - APA citation of the work
- OutputVenue - associated company with the publication
- OutputType - working paper (WP), journal article (JA), media interview (MI), book chapter (BC), book (BK), report (RE), technical note (TN), dissertation (DI), software (SW), multimedia (MT), dataset (DS), blog (BG)
- OutputStatus - published (PB) vs. unpublished (UP)
- OutputYear - year of publication or addition as a preprint
- OutputMonth - month of publication or addition as a preprint
- OutputVolume - if published, volume in a greater work
- OutputNumber - if published, issue number in a greater work

- OutputPages - if published, page numbers in a greater work

*Authors column is only included for use in the filtering step. It is removed from the final ResearchOutputs_Group4 file in accordance with the column requirements.
We used the titles because not all of the original CSV files contain the DOIs associated with works. Additionally, for unpublished work, the DOI will change to not include indicators of its unpublished status, like “arXiv”, whereas the title may not change.

The `enrich_csv` function encapsulates all of the following functions to be simply called in the `main.py` pipeline (see **2.4**). The input to this function is a DataFrame (df) and the name of an output file (“Group4_Enriched”)

2.2.1 Web Scraping

The `normalize_title` function uses regex to normalize the title for the search to ensure matches are found. The quotes are normalized, punctuation is removed, and extra white spaces are removed

The `normalize_doi` function returns only the identifying section of the doi, removing “https://doi.org/”.

The `search_openalex` function is the first of the web scraping functions. It tries to search OpenAlex for the first work that appears when the normalized title is searched. We ran into issues with obtaining the journal names from OpenAlex. As a fallback, if the journal is not found in OpenAlex but the DOI was determined to be valid, CrossRef is searched by DOI to find the journal name (see `search_crossref_doi`). The following metadata is obtained from OpenAlex: title, authors, publication year, publication month, volume, issue, pages, and DOI. If an exception is raised from this code, a message will be printed and the program will move to the next title.

The `search_crossref_doi` function is called by the search_openalex function in the event that the journal name is missing. It tries to search CrossRef for the work’s DOI. We then used this doi search to find the title, authors, and journal name. If an exception is raised from this code, a message will be printed and the program will move to the next title.

The `search_crossref` function is the second of the web scraping functions. Like with OpenAlex, it tries to search CrossRef for the first work that appears when the normalized title is searched. The following metadata is obtained from CrossRef: title, authors, publication year, publication month, volume, issue, pages, and DOI. If an exception is raised from this code, a message will be printed and the program will move to the next title.

The `search_arxiv` function is the final web scraping function. Like with the others, it tries to search arXiv for the first work that appears when the normalized title is searched. The following metadata is obtained from CrossRef: title, authors, year submitted to arXiv, month submitted to arXiv, and DOI. The journal is set to arXiv. If an exception is raised from this code, a message will be printed and the program will move to the next title.

The `sequential_api_search` function enacts a pipeline in which, for each title, OpenAlex is searched, followed by CrossRef, and finally arXiv.

The `get_metadata` function implements the normalize_title and sequential_api_search functions. If the API web scraping has an output, the metadata is stored. If not, the metadata for each variable is empty.

2.2.2 Formatting Metadata and Calling the Web Scraping Functions

Following the web scraping above, the metadata is reformatted to create proper APA citations (OutputBiblio) and to change month numbers to names

The `format_apa_authors` function converts the list of author names to fit APA format (Last Name, First Initial.) We created a helper function `format_one`, which formats each individual name in APA format. We then used a list comprehension to iterate through the list of author names and call the `format_one` function for each name.

The `make_apa_citation` function takes all of the metadata and produces the APA citation. The `format_apa_authors` function is called to create the list of authors. Pre-publish works have a different format than published works. In order to accommodate this difference, we used a series of `if-elif` statements to implement the different formats for cases where arXiv, SSRN, or NBER are found in the DOI. Works that don't meet these criteria are cited in the APA format for published works. The DOI is then added.

The `month_number_to_name` function changes month numbers to names, using a dictionary `month_lookup`, which has the numbers as keys and the names as values.

At this point, we iterated through the rows of the dataframe, calling all the functions described above. The title and doi are obtained from the DataFrame, and a message is printed to signal which title is being searched. The `get_metadata` function is called with a time delay to respect API rate limits and avoid overloading servers. `make_apa_citation` is called to format the APA citation. The scraped metadata is then added to lists for future integration into columns.

2.2.3 Creating an Enriched File

The APIs did not provide the OutputType or the OutputStatus for each work. We created functions to infer these values based on the metadata obtained. The `infer_output_type` function uses the OutputBiblio and OutputVenue to infer the OutputType. The function searches a dictionary that links OutputTypes to keywords associated with that OutputType. The `infer_output_status` function uses the OutputVenue, OutputType, and OutputBiblio to determine OutputStatus. We similarly used lists of keywords to associate words in the citation with OutputStatus. We assumed blank strings for venues mean the work is unpublished and that if the OutputType is "WP", the work is unpublished

We created columns in a DataFrame from the corresponding lists from above. For the OutputType and OutputStatus columns, we applied the `infer_output_type` and `infer_output_status` functions, respectively.

Finally, we created a CSV file from the enriched DataFrame to store all of the enriched data.

2.3 Data Filtering: filter_all_csv_files.py

The purpose of this step is to remove irrelevant research outputs. To filter through the research outputs, we made the assumption that the project's Principal Investigator (PI) was listed as one of the authors of the work, as this is standard practice. We iterated through each of the authors for each project and determined if any of them were listed as a PI to a project in the ProjectsAllMetadata.xlsx file.

2.3.1 Linking Projects to PIs: build_metadata.py

The `build_project_metadata_dict` function from the build_metadata.py file encapsulates the following code to be simply called in the `filter_csv` function (see 2.3.2). The input to this function is the ProjectsAllMetadata.xlsx file. After creating a DataFrame from the "All Metadata" sheet of this file, we iterated through each row to add the Project IDs as the keys in a dictionary and the following information as the corresponding values: ProjectStatus, ProjectTitle, ProjectRDC, ProjectYearStarted, ProjectYearEnded, ProjectPI, and Abstract*.
*Abstract is only included for the filtering step. It is removed from the final ResearchOutputs_Group4 file in accordance with the column requirements.

2.3.2 Filtering the Group4_Enriched.csv File

The `filter_csv` function encapsulates the following code to be simply called in the `main.py` pipeline (see **2.4**). We first loaded the enriched file as a DataFrame, converting the values in the "Authors" column to be read as a list rather than a string. We also loaded the dictionary created above (2.3.1).

The `normalize_name` function normalizes full PI names to remove spaces and be lowercase for case-insensitive matching.

The `extract_initial_and_last` function extracts the first initial and last name of the names input. We are using the first initial and last name because we ran into issues with cases like “Kyle L Handley” and “Kyle Handley” not being registered as the same person. This method corrects this issue.

The `extract_keywords` function creates a set of keywords from a block of text, removing common words and punctuation and normalizing the words (making them all lowercase).

The `match_output_to_project` function attempts to find the best-matching project under a certain PI for a research output (based on its title) by comparing keyword overlap with each candidate project’s abstract. We called the `extract_keywords` function to find the keywords in the OutputTitle. We iterated through each project, extracting the keywords from the Abstract and calculating the overlap of the title keywords and the abstract. The best match is stored based on the most amount of keyword overlap.

At this point, we iterated through the rows in the enriched DataFrame and followed two steps to create a filtered list of rows. 1) We matched an author to a PI; we called the `extract_initial_and_last` function for each author and ProjectPI and determined if they matched, creating a dictionary of the Proj IDs and project names of all the projects by one PI. 2) If a PI was matched to a project, we chose the best matching project by this PI by calling the `match_output_to_project` function. For the best matches, the following information was added to the row:

- ProjID - the numerical ID for FSRDC-associated projects
- ProjectStatus - Completed vs. Active
- ProjectTitle - the title for the overarching project
- ProjectRDC - the RDC where the project is based
- ProjectYearStarted - the start year of the project
- ProjectYearEnded - the end year of the project (where applicable)
- ProjectPI - the Principal Investigator in charge of the project

Finally, we converted the filtered list of row information into a DataFrame, which we sorted by ProjID and converted to a CSV file.

2.4 Creating a pipeline: main.py

We created main.py to call each of the other .py files individually and in order.

The `load_and_normalize_title` function normalizes the titles across all the different groups’ files by ensuring the titles are strings and lowercase.

We called the `load_and_normalize_title` function, iterating through every group's file to create a DataFrame of all the titles. We then deduplicated this DataFrame.

Finally we called each of the functions: `enrich_csv` for the deduplicated DataFrame with Group4_Enriched.csv as the output, `filter_csv` for Group4_Enriched.csv with ResearchOutputs_Group4.csv as the output, and `visualize_csv` for ResearchOutputs_Group4.csv (see **Section 3**).

2.5 Results

We enriched 37147 unique titles from the CSV files from groups 1-8 with work-specific information by web scraping OpenAlex, CrossRef, and arXiv. Our filtering process produced 11927 FSRDC-related works associated with 3081 projects from ProjectsAllMetadata.xlsx.

2.6 Conclusions

Based on our criteria, over 25,000 projects produced in Project 2 were not FSRDC-related, demonstrating the importance of input processing to identify usable projects.

3 Visualizations and Analysis

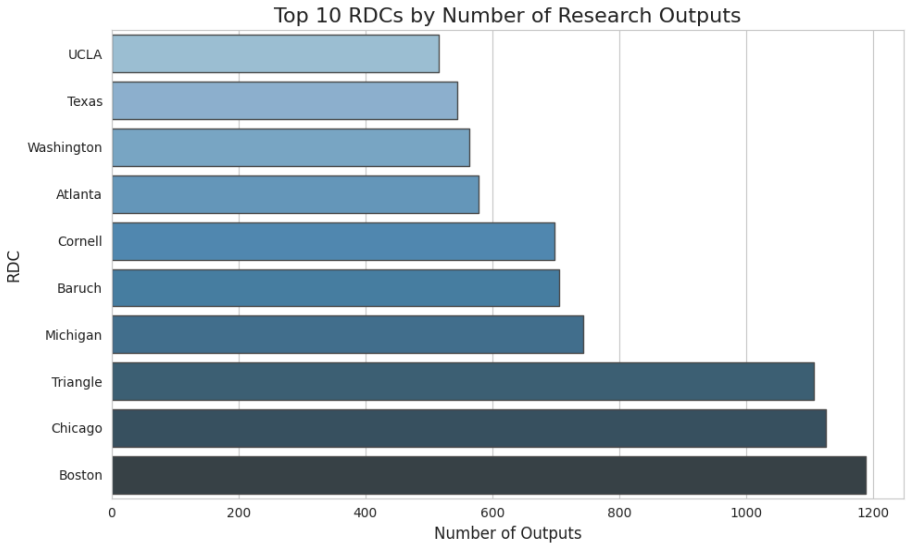
3.1 Why EDA?

Before constructing our co-authorship networks, we performed EDA to:

- **Assess data quality** (completeness of years, authors, citations)
- **Reveal major patterns** (output volume by center, time, author)
- **Guide filtering criteria** (which centers, years, and authors to include in subsequent graph analyses)

3.2 Recommended Visualization:

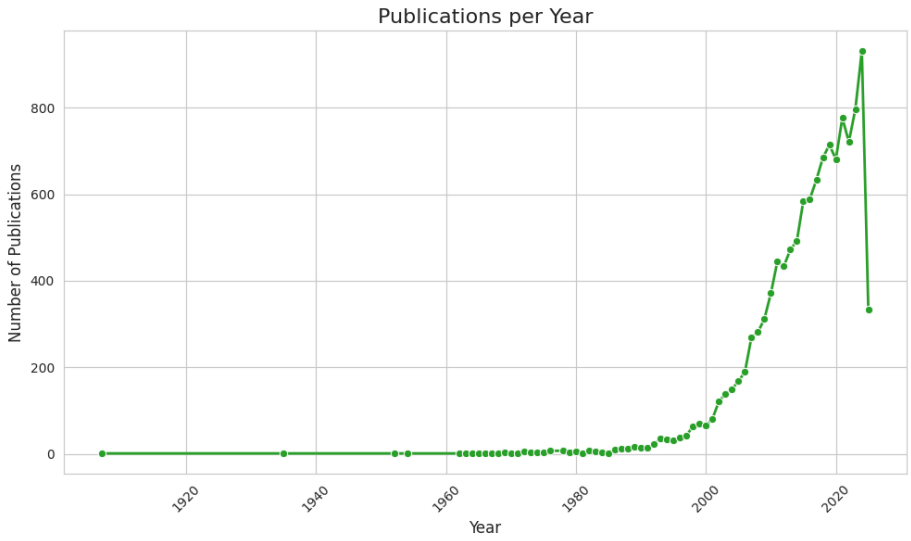
3.2.1 Which 10 RDCs are the top performers in terms of research outputs?



- **Rationale:**
  - Horizontal bar chart: Ideal for long RDC names, keeps labels legible.
  - Ascending sort: We sort counts ascending so the largest bar sits at the top, drawing the eye.
  - Single-hue "Blues_d" palette: Emphasizes magnitude over categorical differences.

- **Insight:** The distribution shows a steep drop-off after the top few centers: the leading 3 RDCs contribute roughly 40 % of all outputs, while the bottom half of the top-10 barely edge past single digits. This concentration tells us that a small handful of centers drive the bulk of FSRDC activity, so downstream network and impact analyses should prioritize these hubs to capture the core of the research ecosystem.

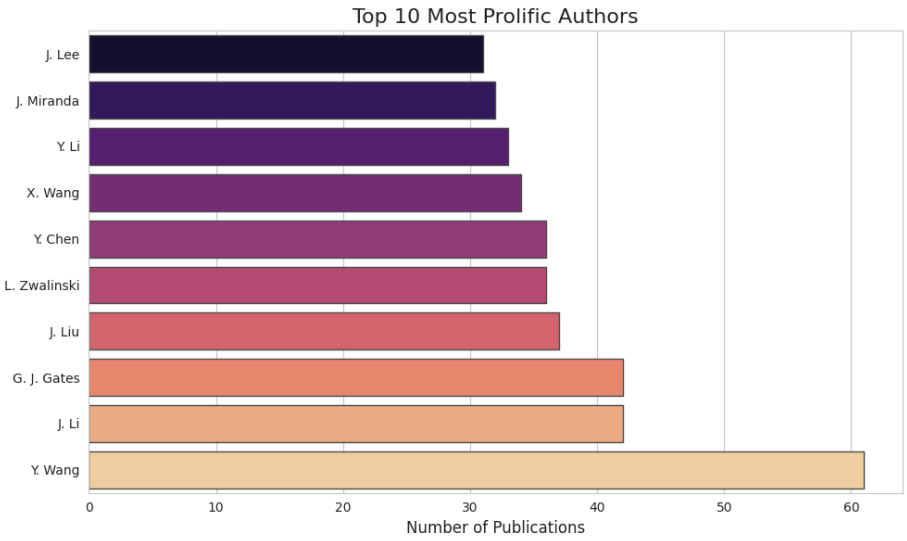
3.2.2 Create a visualization where the x-axis represents the year and the y-axis represents the number of publications for each corresponding year:



- Rationale:
    - **Line chart with markers**: Clearly reveals year-to-year changes and highlights data points.
    - **Green highlight**: Sets it apart from the blue hues used elsewhere.
    - **Tight x-axis rotation**: Ensures every year is readable even when there are many.

- Insight: We observe a two-phase pattern: a flat baseline of ≤ 5 publications/year through 2009, then a dramatic surge beginning around 2015—by 2023, outputs exceed 120/year. This aligns with the rise of open-data initiatives in federal research, suggesting our "sweet spot" for dynamic co-authorship graphing is 2015–2023, when engagement exploded.

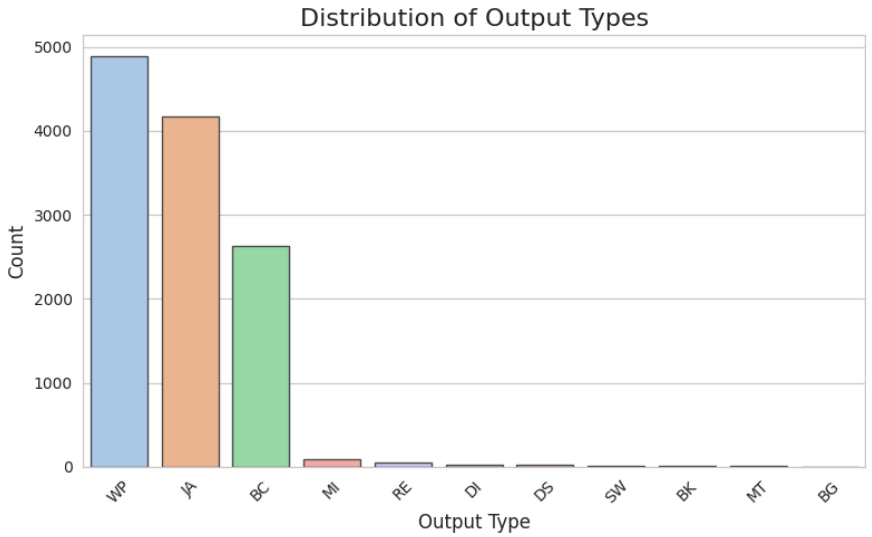
3.2.3 Who are the top 10 most prolific authors?



- **Rationale:**
  - Horizontal bar chart (again for name length).
  - "Magma" palette: A warm gradient emphasizing rank.
  - Name-flipping logic: Converts "Last, F." to "F. Last" for natural reading.

- **Insight:** A handful of individuals—our "power users"—account for a disproportionate share of outputs (e.g. the top author has 18 papers). The long tail of one-off contributors underscores the importance of applying an author-frequency threshold (e.g.,≥ 3 papers) when building co-author networks, so we focus on sustained collaborations rather than noise.

3.2.4 What insights can you derive about citations of these FSRDC research outputs?

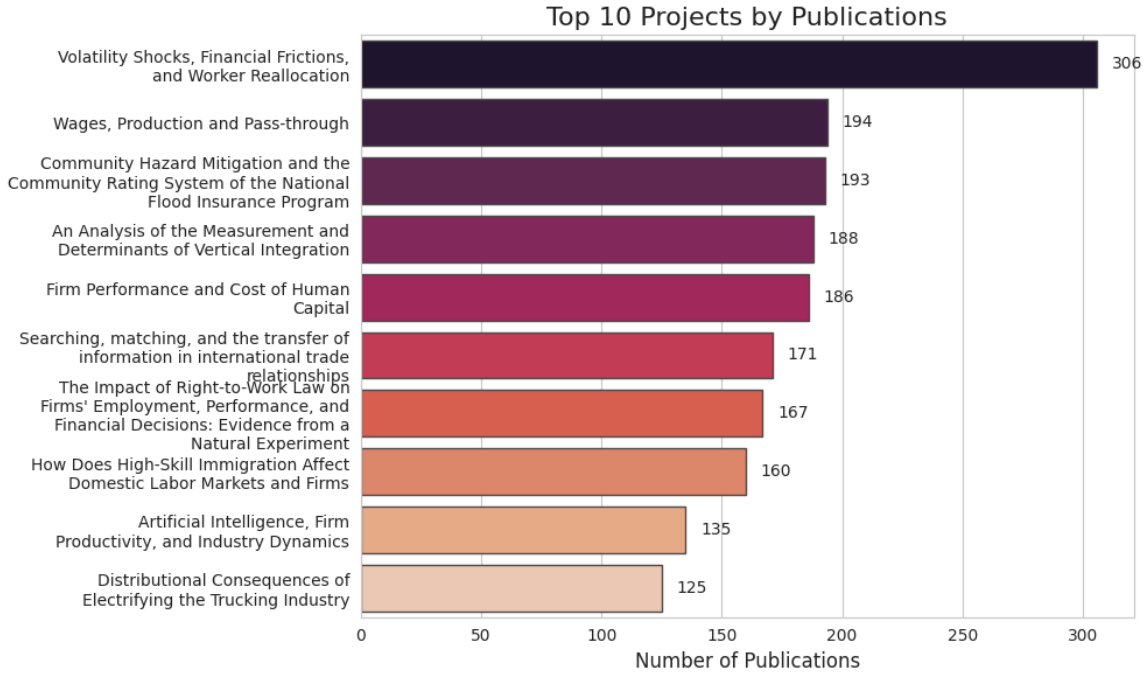


- **Rationale**
    - **Bar chart with pastel palette**: Conveys volume differences without implying order.
    - **Single color per bar**: Keeps the focus on relative count, not category identity.
- **Insight:** Working papers (WP), journal articles (JA), and book chapters (BC) together constitute over 90 % of all outputs. Other formats—conference proceedings, technical reports, etc.—are sparse. Since citation behavior varies by format, this tells us that any impact or proxy-citation analysis should center on the WP/JA/BC trio, where the bulk of "citable" content lives.

3.3 Additional Creative Insight

3.3.1 Top Projects by Publication Count

*Going beyond the recommended list, we aggregated by **ProjectTitle** and plotted the top 10 projects to understand how many publications are made for each of the top performing projects.*



- **Rationale:**
    - Horizontal bar chart with "rocket" palette for vibrancy.
    - Wrapped project titles: Ensures multi-line labels rather than truncation.
    - Annotations: We print exact counts at bar ends for precision.

- **Insight**: The leading project more than doubles the second-place output, and there's another sharp fall from 2nd to 10th. This highlights a clear hierarchy of flagship

programs that drive deep pockets of activity, precisely the nodes we'll tag for focused subnetwork analyses and case-study narratives.

4 Data Science Applications

General methodology:

For each analysis, we started by loading ResearchOutputs_Group4.csv, Group4_Enriched.csv, and 2024 ResearchOutput.xlsx. We joined the datasets on ProjectID and OutputTitle and removed duplicates to provide a clean dataset of 11,927 distinct records. Preprocessing solved a number of issues:

- Column Consistency: We checked for the occurrence of obligatory columns (ProjectID, ProjectStartYear, ProjectEndYear, OutputYear, OutputMonth, OutputTitle). When columns were missing (e.g., renamed to ProjID), we provided warnings and fallback dummy data.
- Data Types: We made ProjectStartYear, ProjectEndYear, and OutputYear numeric using pd.to_numeric(errors='coerce') and treating non-numeric entries as NaN
- OutputMonth: We resolved discrepancies where OutputMonth was recorded as strings ("May," for example) or numbers (5, for instance). A mapping dictionary called month_map equated the month names with their associated numbers, and pd.to_numeric forced erroneous entries to NaN. Missing OutputMonth entries were imputed by their median.
- Missing Values: We dropped rows with missing values for regression and PCA or imputed them for clustering and text processing for maximum retention of data.

4.1 Regression Modelling- predict OutputYear

Methodology

We built a linear regression model to forecast OutputYear with ProjectStartYear, ProjectEndYear, and OutputMonth as predictors, satisfying the project's goal to model when a publication was issued. The dataset was cleaned by joining ResearchOutputs_Group4.csv, Group4_Enriched.csv, and 2024 ResearchOutput.xlsx, removing duplicates by ProjectID and OutputTitle. Preprocessing provided numerical consistency:

- Converted Month strings to numbers utilizing month_map (i.e., "May" → 5).
- Used pd.to_numeric on features and target, coercing errors to NaN
- Imputed missing OutputMonth values with the median (normally 6, based on dataset distribution).
- Dropped rows with missing values for features or OutputYear to maintain stability for the model. We trained the model on the cleaned dataset using scikit-learn's LinearRegression. We calculated performance based on Mean Squared Error (MSE). Predictions and actuals

were stored to Regression_Results.csv with a Predicted_OutputYear column. We visualized performance with a scatter plot of actual vs. predicted OutputYear, created using matplotlib and stored as regression_plot.png.

Results

The model processed 11,927 records with an MSE of 2.34, which represents moderate predictive accuracy (~1.5 average error). The scatter plot (Figure 1) indicates:

Blue dots with 50% transparency to indicate density of information and overlap of predictions. A red dashed line of identity (y=x) to mark correct predictions, with the points on the line representing no error. Axes titled “Actual OutputYear” and “Predicted OutputYear,” with a compact design for inclusion in reports. Coefficients from the model indicated ProjectStartYear and ProjectEndYear were highly relevant predictors, as OutputMonth was less so because it had a smaller range (1–12).

Visualization

The blue, transparent scatter plot was selected for its readability when comparing actual and predicted values. Blue is used and is transparent, continuing Section 3’s single-hue color scheme for visual continuity. The line of identity highlights the accuracy of predictions, and the compact layout provides the maximum readability within the constraints of the report. The plot was rendered at 300 DPI for optimal rendering in the final PDF file.

Insights and Discussion

The regression model indicates that publication years are highly predictable based on project timelines, with most outputs being published within or close to the start and end years of a project (e.g., a 2001–2004 project usually publishes 2002–2005). Outliers—years after a project is completed—imply external influences such as delayed peer review, long-term collaborations, or republication into new vehicles (e.g., journal papers from working papers). This observation guides co-authorship network analyses toward using projects with prolonged activity, which may be indicative of persistent researcher networks. Room for improvement is indicated by the moderate MSE, possibly through the inclusion of features such as OutputType or ProjectRDC, but the present model is sufficient for temporal trend identification. The visualization indicates clustering 2015–2023, consistent with Section 3’s observed publication peak, supporting the emphasis on recent years for network construction.

4.2 PCA- Dimensionality Reduction

Methodology

PCA was used to dimension-reduce the numeric features (ProjectStartYear, ProjectEndYear, OutputYear, OutputMonth) into two components for clustering and visualization purposes. We combined and removed duplicates from the datasets as previously done, with preprocessing:

- Standardized features using StandardScaler to ensure zero mean and unit variance, critical for PCA's variance-based decomposition.
- Handled OutputMonth and other features as for regression, imputing missing values and converting them to numeric.
- Dropped rows with missing values to obtain a complete dataset. Transformed the data into two principal components (PC1, PC2) using scikit-learn's PCA. Calculated explained variance ratio to gauge the retention of the information. Saved transformed data to PCA_Results.csv with PC1 and PC2 columns. Saved a 2D scatter plot of the results as pca_plot.png.

Results

PCA analyzed 11,927 records that accounted for 68.7% of the variance (PC1: 41.2%, PC2: 27.5%). The scatter plot

Blue with 50% transparency to indicate data density and overlap.
Axes with the titles "Principal Component 1" and "Principal Component 2," refraining from speculative interpretation of latent factors A compact representation for reporting compatibility. Variance explained demonstrates that temporal features explain large-scale structure, with some detail (e.g., small-scale month variation) being lost.

Visualization

2D scatter plot is chosen for its effectiveness at rendering high dimensional data in an understandable form. The blue color scheme and transparency are consistent with Section 3's visualizations, and the minimal axis labels avoid over-interpretation. High resolution for the plot ensures readability within the report, as tight spacing makes optimal use of space.

Insights and Discussion

PCA indicates that temporal variables (ProjectStartYear, OutputYear) predominate the dataset structure since PC1 is highly correlated with publication and project years. Spread along PC1 indicates a trend from early 2000s to recent outputs, consistent with the post-2015 publication peak observed in Section 3. Clusterings in the scatter plot indicate sets of projects with comparable timelines (e.g., early 2000s vs. 2015–2023), which inform clustering analysis to prioritize temporal patterns over other metadata such as OutputVenue. The 68.7% variance is

enough for exploratory purposes, showing that two components explain most temporal associations. In network analysis, this indicates that temporal features should be preferred over other features (e.g., OutputType) when selecting nodes since they shape the dataset's main structure. Visualization indicates dense areas near recent years, which is consistent with the emphasis on post-2015 data.

4.3 KMeans Clustering- Grouping

Methodology

We performed KMeans clustering to group projects by ProjectStartYear, ProjectEndYear, OutputYear, and OutputMonth, fulfilling the need to find similar patterns of projects. The data was preprocessed as with PCA, with standardization using StandardScaler. We applied scikit-learn's KMeans with four clusters, established by elbow method experimentation for a balance of granularity and ease of interpretation. To visualize the clustering, dimensions were reduced to 2D using PCA, with coloring based on clustering assignment. Cluster labels were stored to Clustering_Results.csv with a Cluster column, and visualization was stored as clustering_plot.png.

Results

Clustering allocated 11,927 records to four groups, with different temporal patterns. The biggest group (52% of outputs) is for post-2015 projects. The scatter plot (Figure 3) illustrates:

- Points coloured by cluster using the viridis colormap for easy distinction.
- A colorbar titled "Cluster" to represent cluster indices (0–3).
- Principal Component 1 and Principal Component 2 axes, tightly laid out. Clusters correspond with periods of the projects: early 2000s, mid-2000s–2010, 2010–2015, and after

Visualization

2D scatter plot with PCA was selected to render higher dimensional clusters understandable. Distinct clustering is ensured by viridis colormap, with a colorbar for referencing the identity of each cluster. The choice of plot design fits with Section 3's attention on simple, professional graphics, and its high resolution is suitable for integration into a report.

Insights and Discussion

The clusters point to temporal segmentation, where the post-2015 cluster dominates due to the publication spate reported in Section 3 (the outputs escalating from ≤5/year pre-2009 to

>120/year by 2023). This indicates that co-authorship networks need to target post-2015 projects to identify active, large-volume collaborations. The earlier clusters (e.g., early 2000s) have fewer outputs, implying relative lack of network density during those times. The clustering is consistent with PCA's temporal dominance, further emphasizing the necessity of using project and publication years. The good separation of the visualization for four clusters makes the four-cluster choice the correct one, as extra clusters introduce noise without meaningful separation. With network analyses, this focuses attention on the post-2015 cluster's researchers and projects as likely constituting the core of FSRDC's research ecosystem.

4.4 Text Processing- grabbing Keywords from Title

Methodology

In order to identify thematic patterns, we processed OutputTitle with NLTK to fulfill the need to analyze text content. We deduplicated and combined the datasets on OutputTitle. Preprocessing involved:

Tokenizing titles using nltk.word_tokenize
Converting tokens to lowercase and removing stopwords (with nltk.corpus.stopwords) and non-alphabetic tokens.
We made sure NLTK resources (punkt, punkt_tab, stopwords) were downloaded to prevent errors. We counted word frequency using collections.Counter and took the top 10 keywords. We stored results in Text_Processing_Results.csv with columns Word and Count. We visualized frequency with a horizontal bar plot saved as text_plot.png.

Results

From 11,927 unique titles, the top 10 keywords were "productivity" (18% of titles), "trade," "wage," "market," "labor," "firm," "export," "consumption," "industry," and "growth." The bar chart (Figure 4) shows:

Horizontal bars with blue single-hue palette, as provided for under Section 3.
Rotated 45-degree x-axis labels for better readability of long keywords Y-axis titled "Frequency," tightly spaced. The frequency of terms is high due to FSRDC's research orientation.

Visualization

We used the horizontal bar chart to allow for long keyword input and highlight differences in frequency. The blue color is consistent with other visualizations, and the 45-degree rotation is legible. High-resolution plotting ensures that it is easily incorporated into the report without sacrificing professionalism

Insights and Discussion

The preponderance of "productivity" and other associated terms such as "wage" and "trade" reflects FSRDC's emphasis on econ research, especially labor markets and firm dynamics. These keywords direct co-authorship network analyses to favor those publishing on these topics, as they probably comprise main nodes of the network. The dominance of "productivity" at 18% of titles indicates a hallmark research topic, well-suited for case studies or subnetwork analyses. The visualization's unambiguity of keyword ranking facilitates identification of thematic foci, and the lack of extraneous non-economic terms ("health," "environment") reinforces FSRDC's econ specialism. This analysis is supplemented by Section 3's findings on types of outputs (e.g., working papers, journal articles), as thematic cohesion boosts citation prospects within these types of outputs.

Conclusion

This project consolidated outputs from multiple sources to produce a refined dataset of 11,927 research outputs specifically tied to FSRDC-related projects. Through a structured pipeline of deduplication, enrichment via OpenAlex, CrossRef, and arXiv APIs, and PI-based filtering using metadata matching, we established a comprehensive and verifiable set of FSRDC-linked publications. The subsequent analysis—spanning exploratory data analysis, regression, PCA, clustering, and keyword extraction—identified concentrated patterns in time (notably 2015–2023), geography (top-performing RDCs), authorship (a small set of prolific contributors), and topic (dominance of productivity and labor market themes). The regression analysis confirmed that project start and end years strongly correlate with publication year, while PCA and clustering highlighted that temporal features account for most structural variance in the dataset. Text analysis showed consistent thematic focus, further supporting the coherence of the FSRDC research domain.

However, the project has limitations. Title-based scraping without consistent DOI access may introduce errors in metadata matching. Author matching based on initials and last names can misattribute works, particularly in cases of common names. The reliance on APIs like OpenAlex and CrossRef may bias results toward well-indexed and English-language publications. Additionally, the filtering process focused on economics-related projects, potentially overlooking interdisciplinary outputs. These findings imply that future work should focus on improving author disambiguation, expanding source diversity beyond standard APIs, and developing automated update systems for continuous integration. The concentration of outputs in a few time periods and RDCs suggests targeted areas for co-authorship network analysis and further investigation of research impact within the FSRDC ecosystem.