BA820 — Project M2

Project Title: Code Trends, Quantified: Mapping the Programming Language Ecosystem

Section and Team: B1 Team 14

Student Name: Vishesh Goyal

1. Refined Problem Statement & Focus

1.1. Business / Domain Question

This milestone examines whether programming languages cluster differently depending on the lens applied - community and adoption (job market presence, user base, visibility) versus technical (file extension ecosystems). The central question is whether technical similarity corresponds to comparable market adoption, or whether mismatches exist between the two views.

1.2. Refinements from M2/M3

The M2 baseline used KMeans on community metrics and a one-hot extension matrix. M4 introduced three key improvements: replacing the Top-30 extension overlap with a TF-IDF Weighted Bag-of-Words representation using cosine distance and hierarchical clustering; adding robustness testing through TOP_N sensitivity analysis and KMeans seed stability checks; and applying the M3 archetype, ghost, and survival tier frameworks to enrich cluster interpretation.

1.3. Assumptions Challenged, Validated, or Invalidated

The assumption that technical similarity implies comparable adoption was challenged -alignment metrics consistently showed this not to be the case. The assumption that community clustering would cover the full dataset was also challenged, as missing GitHub and Wikipedia values limited it to a smaller high-coverage subset. The use of log transformations for heavy-tailed metrics was validated, yielding stable and interpretable clustering results.

2. EDA & Preprocessing: Updates

2.1. Carried Forward from M2

The core analytical structure from M2 was retained: defining a community-based view using demand and visibility metrics, and a technical view using file-extension ecosystems. Given that these metrics exhibit heavy-tailed distributions, $\log_{10}(x+1)$ transformations are applied prior to clustering and visualization.

2.2. Changes and Revisions in M4

2.2.1. Unified preprocessing: Numeric casting and log-transformed feature creation were standardized so that both the community and technical analyses operate on consistent inputs. Additionally, negative Wikipedia view values were treated as missing rather than valid data points.

2.2.2. Explicit missingness accounting: Community clustering requires complete observations across four key features. After removing rows with missing values, only 88 records remain in the working dataset. This reduction is driven primarily by two features with substantial missing rates - GitHub repository stars (79.34% missing) and Wikipedia daily page views (67.28% missing) - while user and job-related fields are fully populated within this subset. This limitation is explicitly acknowledged in the analysis.

2.2.3. Extension parsing for the technical view: GitHub language file extension data was parsed and cleaned to produce a list of extensions per language. Only languages with at least one valid extension were retained, yielding a working set of 466 languages for the technical clustering analysis.

3. Analysis & Experiments

3.1. View A - Community Clustering

Community clustering was performed using KMeans applied to four standardized log-transformed features: $\log_{10}$ number of users, $\log_{10}$ number of jobs, $\log_{10}$ GitHub repository stars, and $\log_{10}$ Wikipedia daily page views. The analysis was conducted on the 88 languages that had complete data across all four features.

This approach is appropriate because community-based segmentation is expected to surface market-facing tiers, distinguishing mainstream languages from those in the long tail. A silhouette sweep across k = 2 through 8 was used for cluster selection, with k = 2 yielding the best silhouette score of 0.6255, followed by k = 3 at 0.4340. Both PCA and UMAP projections supported this result, showing clear two-group separation. The key takeaway is that when all four community signals are jointly available, they produce a well-defined two-tier structure. An important caveat, however, is that these results reflect a high-coverage subset of the data rather than the full dataset.

3.2. View B - Technical Clustering Baseline

The technical clustering baseline was built using a Top-30 one-hot extension matrix - 30 binary columns indicating the presence of each of the most common file extensions - supplemented by a feature capturing the total number of extensions listed per language. After standardization, KMeans was applied to this 466 × 31 feature matrix, covering all languages with at least one parsed extension.

This approach is appropriate because extension footprints are expected to group languages into technical ecosystems defined by shared tooling and file-type conventions. A silhouette sweep from k = 2 to 8 identified k = 3 as optimal, with a silhouette score of 0.8109, though k = 2 performed nearly as well at 0.8070. The consistently high silhouette scores across values of k indicate a robust technical structure that is not overly sensitive to the choice of cluster count.

3.3. New in M4 - Method Upgrade for Technical Ecosystems

3.3.1. Dead End and Adjustment:

An initial Jaccard-based approach was explored but proved unstable during cluster selection, with k drifting inconsistently and producing fragmented, disproportionately small clusters. This was replaced by a token vector representation using cosine distance, chosen for its greater stability and interpretability.

3.3.2. New Method:

File extensions are treated as tokens and encoded using TF-IDF weighting (Weighted Bag-of-Words). Cosine distance is then computed across all language pairs, and hierarchical clustering is applied to the resulting precomputed distance matrix. This process yielded 99 token features and a 466 × 466 distance matrix.

3.3.3. Cluster Selection:

Silhouette scores were evaluated across k = 2 through 25, with an additional constraint enforcing realistic minimum cluster sizes to prevent fragmentation. The best defensible solution was k = 2, achieving a silhouette score of 0.6947 with no evidence of cluster fragmentation.

3.3.4. Comparison to Baseline:

The Adjusted Rand Index (ARI) between the baseline technical KMeans clustering and the new hierarchical method is 0.0612, indicating that the two representations produce meaningfully different segmentations. This confirms that the choice of technical representation has a substantial impact on clustering outcomes.

3.3.5. Interpretability Enhancements:

As a key M4 improvement, the top tokens and representative example languages were identified for each cluster, enabling

the ecosystems to be meaningfully named and interpreted beyond raw cluster assignments.

3.4. Sub-Q3 - Alignment vs. Mismatch (Core Question for M4)

3.4.1. Approach: Languages were matched across the two views using a shared identifier (pldb_id), and all alignment analysis was restricted to the resulting overlap set of 52 languages for which both technical and community data were available.

3.4.2. Evidence for Alignment and Mismatch: Three forms of evidence were used to assess the relationship between the two views. First, a row-percentage alignment heatmap was constructed to visualize how languages within each technical cluster distribute across community clusters. Second, Adjusted Rand Index (ARI) scores were calculated as a quantitative alignment check: the baseline technical-to-community ARI was 0.0000, while the new hierarchical method improved this slightly to 0.1547 - though both values indicate weak overall agreement. Third, specific mismatch examples were identified: languages such as PHP, Swift, Rust, Kotlin, TypeScript, and Julia are grouped together on technical grounds yet fall into the high-adoption, high-visibility community cluster, illustrating cases where technical similarity does not translate to comparable market outcomes.

3.4.3. Conclusion: The analysis reveals that technical and community clusterings are more often misaligned than consistent. While the improved technical representation yields a modest gain in agreement, it does not resolve the fundamental mismatch between how languages cluster technically and how they perform in terms of adoption and visibility.

3.5. M3 Lenses Applied (Deepening Interpretation)

To better explain why languages within the same technical ecosystem can exhibit different adoption behaviours, the M3 interpretive lenses - ghost rate, survival tier, and archetype composition — were applied at the cluster level.

Regarding ghost rate, Cluster 1 shows a higher proportion of ghost languages (approximately 1.1%) compared to Cluster 0 (approximately 0.3%). While the difference is directionally meaningful, ghost languages remain rare across both clusters overall.

On survival tier distribution, Cluster 1 contains a higher share of actively maintained languages (around 78.9%) relative to Cluster 0 (around 69.5%), while Cluster 0 shows a greater proportion of languages classified as Maintained or Dormant.

In terms of archetype composition, Cluster 0 is predominantly characterized by archetype 0 (approximately 50% of the cluster), whereas Cluster 1 is more evenly divided between archetypes 1 and 3, each accounting for roughly 37.5%.

Together, these lenses add meaningful business context - capturing dimensions of maturity and demand or attention style - that technical token representations alone cannot reliably convey, and help explain the adoption-level differences observed between technically similar language groups.

3.6. Association-Style Co-occurrence (Interpretability)

To further support the interpretability of technical ecosystems, association rule mining was applied to the most common file extensions, using support, confidence, and lift as evaluation metrics. This approach identifies extension pairs or bundles that tend to co-occur consistently across languages. Rules with high lift values indicate niche but

strongly coupled extension workflows - that is, combinations that appear together far more often than would be expected by chance. However, these high-lift rules are interpreted with caution, as many are associated with low support values, meaning they apply to a relatively small number of languages and may not generalize broadly.

3.7. Robustness and Reliability

3.7.1. Technical Baseline Sensitivity (TOP_N): Sensitivity testing across three vocabulary sizes showed that optimal cluster count and silhouette scores varied by setting (TOP_N = 20: k = 2, 0.8553; TOP_N = 30: k = 3, 0.8109; TOP_N = 50: k = 2, 0.8860). While the TOP_N = 20 and TOP_N = 30 configurations yielded similar assignments (ARI = 0.7468), TOP_N = 30 and TOP_N = 50 diverged considerably (ARI ≈ 0), suggesting that although a strong technical structure is consistently present, cluster membership is sensitive to the vocabulary definition.

3.7.2. Community Clustering Seed Stability: Across 10 re-runs with varying random seeds, community KMeans produced a perfect ARI of 1.0 in all cases, confirming that the two-cluster solution is fully reproducible.

4. Findings & Interpretations

4.1. Community clustering (log-scaled users, jobs, GitHub stars, Wikipedia views) favored k = 2 (silhouette = 0.6255), producing a stable two-tier structure - a low-adoption long tail (Cluster 0) and a high-demand mainstream segment (Cluster 1) - confirmed by both PCA and UMAP.

4.2. Technical clustering on 466 extension-available languages yielded strong baseline separation (k = 3, silhouette = 0.8109). The M4 TF-IDF Weighted Bag-of-Words upgrade (99 token features, cosine distance, hierarchical clustering) produced meaningfully different segmentation from the baseline (ARI = 0.0612), with k = 2 as the best defensible solution (silhouette = 0.6947), confirming that representation choice materially affects ecosystem groupings.

4.3. Alignment analysis on the 52-language overlap showed predominant mismatch: baseline technical-to-community ARI = 0.0000, improving only marginally to 0.1547 with the upgraded method. Languages such as PHP, Swift, Rust, Kotlin, TypeScript, and Julia - technically co-clustered yet occupying the high-adoption community cluster - exemplify "similar tech, different market outcomes."

4.4. M3 lenses revealed that Cluster 1 is more Active (~78.9%) with a higher ghost rate (~1.1%), while Cluster 0 skews Maintained/Dormant, reflecting maturity differences that partially explain adoption mismatches. Archetype composition also diverged (Cluster 0: ~50% Archetype 0; Cluster 1: split between Archetypes 1 and 3).

4.5. Robustness testing confirmed community clustering is fully reproducible (seed ARI = 1.0), while technical baseline membership is sensitive to vocabulary size (TOP_N shifts optimal k; ARI between TOP_N = 30 and 50 ≈ 0), reinforcing the Weighted Bag-of-Words method as a more stable representation.

Key takeaway: Technical ecosystems cluster cleanly, but alignment with community adoption segments remains weak even after methodological improvements - languages sharing similar extension footprints can occupy vastly different positions in terms of jobs, users, and visibility. This demonstrates that technical similarity is not a reliable proxy for market relevance. For practical decisions such as curriculum design, hiring strategy, or tooling investment, community and technical signals should be treated as distinct axes - the former for market positioning, the latter for ecosystem and compatibility interpretation.

5. Appendix

5.1. GitHub Repository

Link to GitHub Repo: *Click Here*

5.2. Other Important Links

Link to Primary Dataset: *Programming Languages Dataset Link*
Link to M4 Colab Notebook: *M4 Colab Link*
Link to M3 Colab Notebook: *M3 Colab Link*
Link to M2 Colab Notebook: *M2 Colab Link*

5.3. Process Overview

In M4 Q1, languages.csv was loaded and preprocessed through a unified pipeline: key fields cast to numeric, invalid values cleaned (e.g., negative Wikipedia views removed), and log10(x+1) features created for heavy-tailed community metrics. Three analysis populations were defined: a community subset requiring complete users/jobs/stars/wiki data (88 rows), a technical subset parsed from github_language_file_extensions retaining languages with at least one valid extension (466 rows), and an overlap set joined on pldb_id for alignment testing (52 rows).

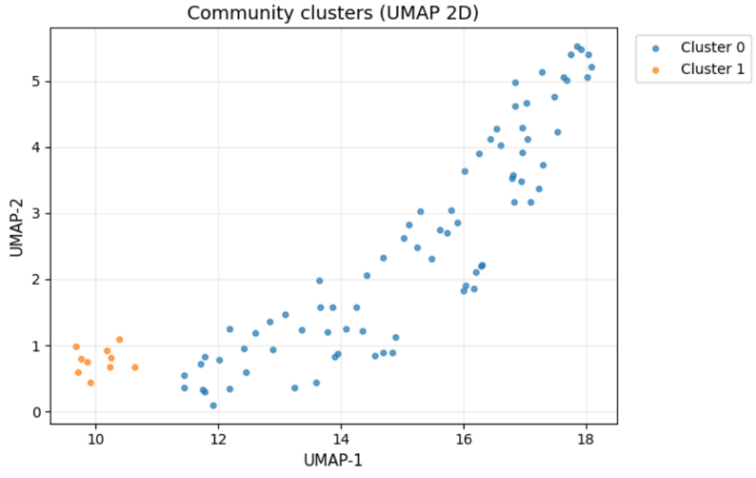
M2 baselines were reproduced by running KMeans on standardized community features (k selected via silhouette, validated with PCA/UMAP, profiled via medians) and building a Top-30 one-hot extension matrix plus extension-count feature for technical clustering. For the M4 upgrade, extensions were represented using Weighted Bag-of-Words (TF-IDF), cosine distances computed, and hierarchical clustering applied with silhouette-based k selection constrained to avoid degenerate clusters; top tokens and example languages were added for interpretability.

Alignment between technical and community clusters was evaluated on the overlap set via a row-% heatmap, ARI, and mismatch examples. M3 lenses — ghost rate, survival tiers, and archetype composition — were applied to contextualize each technical ecosystem's adoption style and lifecycle maturity, supplemented by an association-rule extension co-occurrence table highlighting tightly coupled extension bundles. Robustness checks covered TOP_N sensitivity for the technical baseline and random-seed stability for community KMeans, with final synthesis addressing all sub-questions and changes from M2.

5.4. Supplementary Material

5.4.1. Figure 1: Community Clusters UMAP

UMAP shows two clearly separated adoption/visibility segments, supporting k=2 as the natural community split.



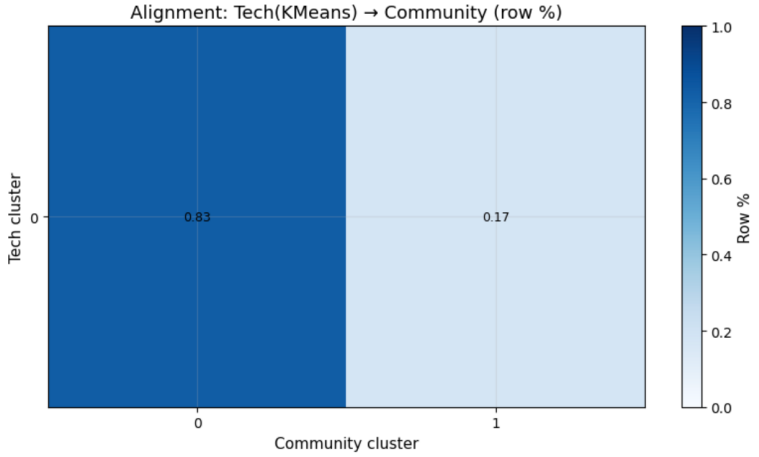
5.4.2. Figure 2: Community Cluster Profile

Median users/jobs/stars/wiki sharply distinguish a long-tail cluster from a high-adoption/high-demand cluster.

comm_cluster	number_of_users	number_of_jobs	github_repo_stars	wikipedia_daily_page_views
0	1120.5	0.0	474.5	24.5
1	173526.5	1239.5	37753.0	1128.5

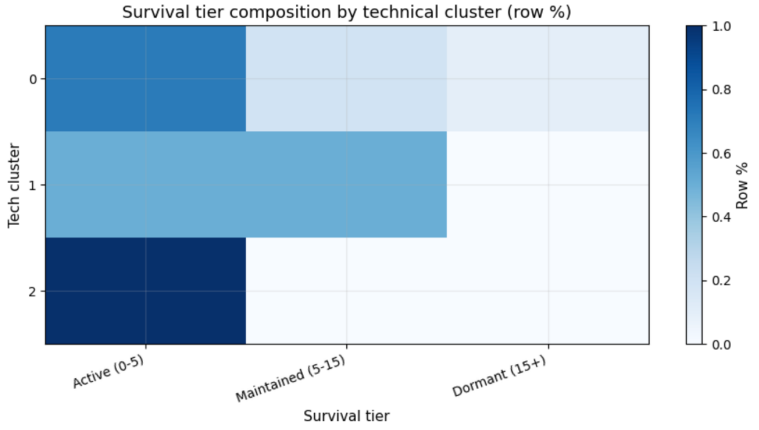
5.4.3. Figure 3: Tech/Community Alignment Heatmap

Row-% heatmap shows partial mapping of technical ecosystems to community segments and reveals mismatch (spillover across segments).



5.4.4. Figure 4: Survival Tier Composition by New Technical Cluster

NEW technical clusters differ in lifecycle maturity: one cluster is more Active while the other has more Maintained/Dormant share.



5.5. Use of Generative AI Tools

I used ChatGPT, Claude AI, and Gemini in Google Colab to support my M4 deliverable. ChatGPT helped me deepen my understanding of technical concepts including TF-IDF weighting, cosine distance, hierarchical clustering, Adjusted Rand Index (ARI) interpretation, and association rule mining metrics (support, confidence, and lift). It also assisted in generating initial code snippets for tasks such as setting up the hierarchical clustering pipeline, computing the co-occurrence rule framework, and building robustness testing loops, which I then adapted, modified, and debugged to fit my specific analysis. Claude AI was used to paraphrase and refine my written findings,

producing concise and formal language for the report narrative and code notebook explanations. Gemini in Colab provided real-time debugging support and syntax assistance during coding sessions.

All core analytical work was completed independently by me, including: defining and refining the technical and community feature sets, making preprocessing and missingness decisions, selecting the TF-IDF and hierarchical clustering approach after identifying the limitations of the Jaccard-based method, running and interpreting clustering experiments across both views, conducting TOP_N sensitivity and seed stability analyses, performing alignment evaluation on the 52-language overlap set, applying M3 lenses to interpret cluster behavior, and drawing conclusions about technical-community mismatch. Generative AI tools were used solely to support conceptual understanding, improve code efficiency, and refine written communication - not to drive any substantive analytical or interpretive decisions.

Link to ChatGPT: *ChatGPT Link*
Link to Claude: *Claude Link*
Link to Gemini: *Gemini Link*

5.6. Special Note

Due to the constraint on number of pages in this report, I had to use Claude and other generative AI tools to compress the text for each section, you can refer to my Colab notebook for the complete analysis (Link in Appendix 5.2)