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 What I am investigating (my M2 focus)

In this milestone, I investigated whether programming languages form meaningful clusters from two perspectives: a technical lens (file extensions + GitHub metadata) and a community/adoption lens (users, jobs, Wikipedia views, GitHub popularity, and temporal signals). If these clusters prove stable and interpretable, they can guide recruiting/training priorities, distinguish market-relevant languages from the long tail, and identify technically similar languages with divergent adoption patterns.

1.2 What changed (or didn't) since M1 — and why?

My core question remained unchanged, but my approach became more realistic. In M1, I planned a single unified clustering approach blending technical and adoption signals across all languages. In M2, I discovered that extension data is missing for 89% of languages, so forcing one feature set would bias results and reduce interpretability. I split the analysis into two feature sets: X_tech (technical subset using only the 470 languages with extension data) and X_comm (community signals across all 4,303 languages). This split improved clarity by allowing each feature set to answer a distinct sub-question without mixing incompatible data coverage.

1.3 Assumption

- **Challenged:** "File extensions are available for most languages." Extension coverage is severely limited, so X_tech must be treated as a subset analysis only.
- **Validated:** "Community signals separate languages into adoption tiers." Community clustering strongly supports this, revealing a small high-adoption group versus a large, long-tail group.
- **New caution:** Missingness is informative for community metrics, but I must carefully distinguish between "low activity" and "missing data."

2. EDA & Pre-Processing: Updates

2.1. M1 findings that mattered most for M2

Two key insights from my initial EDA shaped my M2 approach. First, heavy missingness in popularity columns (stars, Wikipedia views, jobs, users) meant many "zeros" actually represent "unknown" rather than true absence of activity. Second, extreme skew in adoption signals - where a handful of languages dominate - would distort distance-based clustering without transformation.

2.2. What I changed in preprocessing for M2 (and why)

To create more meaningful and fair clusters, I implemented several preprocessing steps. For missing values, I used median imputation for numeric features and mode for categorical features, ensuring models could run consistently without excessive data loss. I applied log transformations to highly skewed community variables, preventing a few extreme languages from dominating distance calculations and making "mid-tier versus high-tier" separations clearer. I also scaled all numeric features so no single metric (like user counts or star counts) could overwhelm the clustering objective.

Most importantly, I built two separate feature matrices instead of forcing one unified approach. X_tech was constructed only for the 470 languages with actual extension data, creating a smaller but technically meaningful subset. X_comm used community and adoption signals across the full 4,303-language dataset, supporting broader market

segmentation analysis. This dual-matrix approach acknowledged data limitations honestly while maximizing what each feature set could reveal.

2.3. What I tried but avoided (dead ends)

I initially considered imputing missing extension lists or inferring technical fingerprints from other metadata. However, I rejected this approach because it would introduce artificial structure and potentially inflate clustering quality metrics with fabricated patterns. Instead, I chose a more defensible strategy: treat X_tech as a smaller, trustworthy subset and explicitly document its coverage limitations rather than compromise analytical integrity.

3. Analysis & Experimentation

3.1. Why clustering is the right tool for my question

My objective is exploratory rather than predictive - I'm not forecasting outcomes but discovering whether natural groupings exist in the data. Unsupervised clustering suits this goal because it reveals whether languages segment into consistent groups, whether those segments are interpretable, and whether technical similarity aligns with market adoption patterns.

3.2. Method 1 — K-Means clustering (primary method)

3.2.1. Why K-Means

K-Means served as my baseline because it's simple, scalable, and produces easily comparable clusters through centroids and medians. Since I standardized all features, distance-based segmentation is appropriate and interpretable.

3.2.2. What I tried (parameters and variants)

I explored multiple k values using an elbow-style approach, selecting k values that balanced interpretability with stability. For X_tech, I chose k=5, while X_comm worked best with k=4. I also generated PCA and UMAP visualizations after clustering - not as clustering methods themselves, but to validate cluster separability visually.

3.2.3. What worked / what didn't

K-Means produced stable, interpretable patterns, particularly for X_comm where adoption tiers emerged clearly. However, K-Means is sensitive to how missingness is represented. In X_comm, some clusters reflect "low or unknown signals" rather than distinct behavioral segments, requiring careful interpretation to avoid overstating findings.

3.3. Method 2 — Hierarchical clustering (Agglomerative Ward) for X_tech validation

3.3.1. Why I used it

For X_tech, I ran Agglomerative clustering with Ward linkage as a robustness check. Hierarchical methods can capture structure even with unbalanced cluster sizes, addressing the question: am I seeing genuine structure or just a K-Means artifact?

3.3.2. Result (agreement with K-Means)

K-Means and Ward clustering showed strong agreement on X_tech (ARI $\approx$ 0.81), confirming that the technical clusters represent consistent, recoverable structure rather than algorithmic noise. Multiple methods converging on similar groupings validates the patterns.

3.4. PCA (Scree + 2D projection) — interpretation and component selection

3.4.1. Scree plot interpretation

I examined scree plots to understand variance distribution and whether a few dimensions dominate the data. For X_tech, PC1+PC2 explain approximately 53% of total variance - a reasonable summary though some structure is lost. For X_comm, the scree plot elbow occurs early, indicating that a few adoption dimensions capture most variance.

3.4.2. Why I used 2 PCA components

I chose 2 PCA components primarily for visualization clarity. Two - dimensional plots provide the most readable way to assess cluster separation. The scree plots show diminishing returns after the first few components, meaning PC1 and PC2 capture dominant structure while maintaining interpretability. My goal wasn't perfect data reconstruction but rather a consistent, comparable 2D view across X_tech and X_comm. If PCA 2D shows weak separation, it signals the need for non-linear embeddings like UMAP rather than adding more PCA dimensions.

3.5. UMAP (2D) — insights and advantages over PCA

3.5.1. X_tech UMAP interpretation

The X_tech UMAP plot reveals multiple compact islands, suggesting technical fingerprints create genuine, discrete groupings of languages sharing similar extension patterns and metadata. I used UMAP over PCA because PCA's linear nature can blur separation when structure is non-linear, while UMAP preserves local neighborhood similarity, making cluster boundaries visually sharper.

3.5.2. X_comm UMAP interpretation

The X_comm UMAP plot displays a dense core surrounded by scattered outliers and mini-groups, consistent with a "long tail plus few high-activity clusters" pattern. UMAP proved more effective than PCA here because adoption signals often form non-linear tiers (core versus extremes), and UMAP renders this tiering more clearly than linear PCA projection.

3.6. Challenges, dead ends, and adjustments

Missingness posed the biggest challenge. Many community signals are absent for large portions of the dataset, requiring careful distinction between "low activity" and "unknown activity" during interpretation. Technical clustering coverage is inherently limited - X_tech is only meaningful for the 470-language subset with extension data, so I explicitly framed it as a subset analysis rather than generalizing to all languages. I intentionally avoided overengineering solutions like complex imputation or synthetic feature generation, as these would inflate apparent quality without being methodologically defensible.

4. Findings and Interpretations

4.1. Key insight 1 — Community features naturally form adoption tiers

Community signals reveal that programming languages organize into distinct market tiers: a compact group of mainstreams, high-adoption languages versus an extensive long-tail of low-activity or unknown-status languages. From a business perspective, this segmentation provides actionable intelligence - organizations making strategic decisions about curriculum development, hiring strategies, or tooling investments can identify which languages demonstrate consistent market demand and sustained attention. This clustering effectively distinguishes languages worth prioritizing from those in niche or declining segments.

4.2. Key insight 2 — Technical similarity exists but is constrained by data availability

Technical clustering produces interpretable, stable results (evidenced by strong agreement between K-Means and Ward methods), but applies exclusively to the 470 languages with available extension data. In practical terms, technical clusters are valuable for ecosystem-level discussions - identifying languages that share tooling compatibility, development environments, or syntax patterns - but cannot serve as universal segmentation across the entire dataset. The limited coverage means technical insights are reliable within their scope but shouldn't be generalized beyond the subset with complete extension fingerprints.

4.3. Key insight 3 — PCA vs UMAP: different roles

PCA and UMAP serve complementary purposes in my analysis. PCA provides rapid assessment of dominant structure through scree plots and establishes a consistent 2D baseline for comparison across feature sets. UMAP enhances visualization when underlying relationships are non-linear, revealing cluster separation that PCA's linear projections might obscure. Using both methods together safeguards against falsely attributing structure that emerges only from a single visualization technique, ensuring my interpretations reflect genuine data patterns rather than methodological artifacts.

4.4. Key insight 4 — Business Implication

The community-based clustering reveals that programming languages naturally segment into distinct adoption tiers—a small mainstream group with high activity versus a large long-tail with low or unknown engagement. Companies can maximize ROI by prioritizing hiring, training, tooling, and integrations around the mainstream cluster rather than distributing resources equally across thousands of languages.

5. Next Steps

5.1. What is not done yet

Several analyses remain incomplete. I have not addressed Sub-Question 3, which examines alignment and mismatches between technical clusters (X_tech) and community clusters (X_comm). Deeper interpretability work—such as naming clusters based on distinguishing features like characteristic extensions or adoption signals—has not been completed. Additionally, I have not explored density-based clustering approaches (DBSCAN/HDBSCAN); my current analysis focuses exclusively on K-Means with Ward clustering for X_tech validation.

5.2. What I will do next (M3 / Phase 3 plan)

For M3, I will develop detailed cluster profiles reporting cluster size, representative languages, and key summary statistics (median log1p values for users/jobs/Wikipedia; most common extensions) to translate numeric labels into plain-English descriptions.

I will conduct stability checks by varying k slightly ($k \pm 1$) and adjusting UMAP parameters. To address high missingness in community metrics, I'll test alternatives like adding missingness indicators or analyzing a higher-coverage subset, ensuring clusters reflect genuine patterns rather than "unknown versus known" signal artifacts.

Finally, for Sub-Question 3, I will compare technical versus community cluster assignments using cross-tabulation and agreement metrics (ARI/NMI), highlighting meaningful mismatches—languages that are technically similar but occupy different adoption tiers, or vice versa.

6. Appendix

6.1. GitHub Repository

Link to GitHub Repo: *Click Here*

6.2. Other Important Links

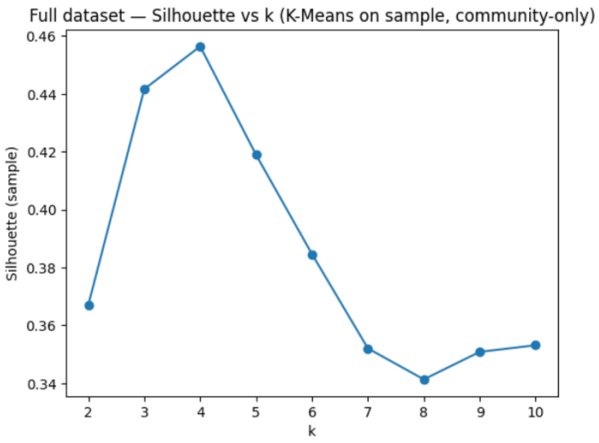
Link to Primary Dataset: *Programming Languages Dataset Link*
Link to Main Code File: *Colab M2 Q1 B1 Team 14*
Link to EDA M1*:* *M1 EDA*
Link to M1 Submission Notebook: *M1 Submission*

6.3. Process Overview

For this M2 phase, I analyzed 4,303 programming languages using unsupervised machine learning to see if they naturally cluster into meaningful groups. I split my analysis into two parts: Feature Set A (X_tech) focused on technical fingerprints like file extensions and GitHub metadata, but I could only use 470 languages here since extension data was missing for about 89% of the dataset. Feature Set B (X_comm) looked at community signals like GitHub stars, Wikipedia views, number of users, and job postings across all 4,303 languages. After cleaning the data with imputation, log-transforms, and standardization, I ran K-Means clustering as my main method and used PCA and UMAP for visualization plus Agglomerative Clustering to validate my results. The technical clustering gave me 5 groups with strong agreement between methods (ARI=0.81), while the community clustering produced 4 clear adoption tiers, including a small but distinct cluster of 80 mainstream, high-adoption languages.

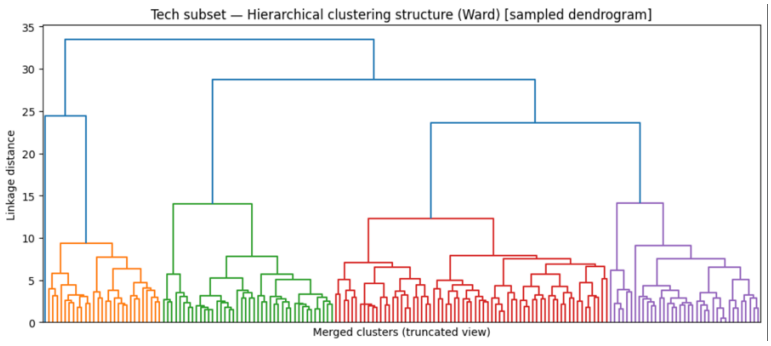
6.4. Supplementary Material

6.4.1. Silhouette Method to find the value of K for clustering of X_comm, here the value of K is 4 as it is the highest peak.



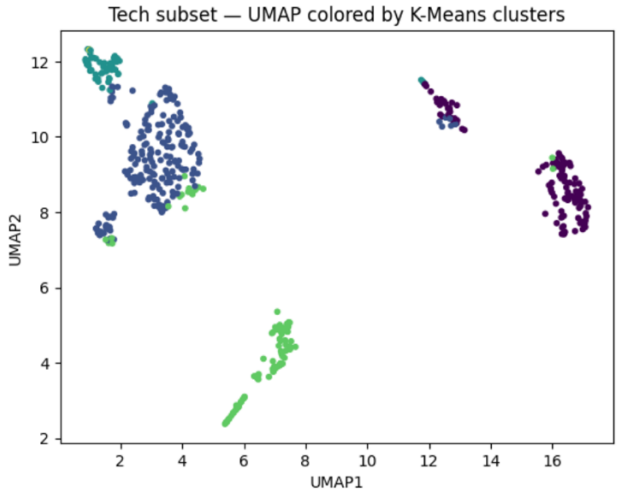
6.4.2. *Following is the dendrogram for Hierarchical clustering structure on X_tech*

This dendrogram illustrates hierarchical merging in X_tech: shorter heights indicate similar languages, while taller merges show distinct groups. The large jumps near the top suggest well-separated major clusters, justifying the chosen cut height for stable groupings.

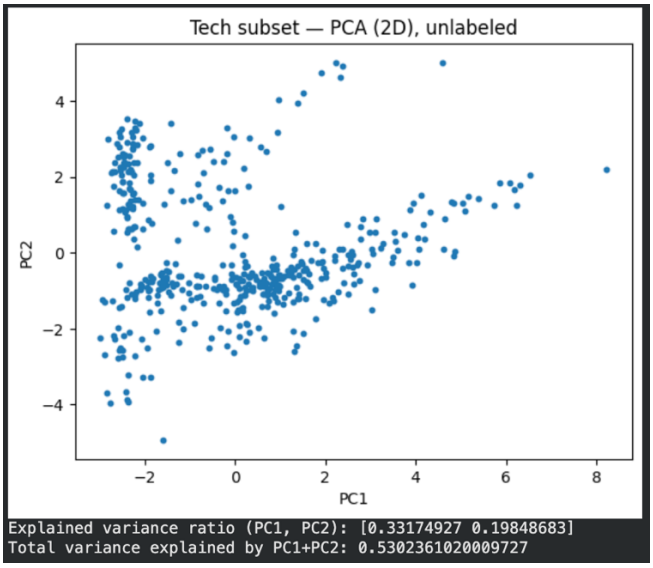


6.4.3. *X_tech UMAP after K Means Clustering*

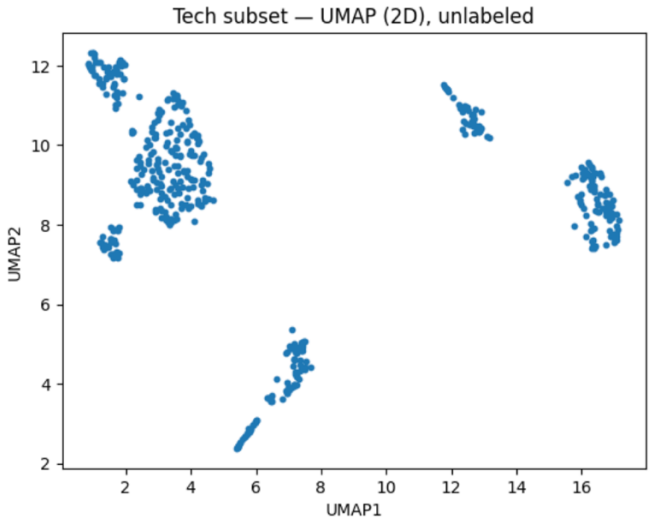
The UMAP embedding highlights clearly separated K-Means clusters in X_tech, suggesting structured technical groupings among languages, where dense clusters represent mainstream patterns and isolated clusters reflect atypical feature profiles.



6.4.4. *X_tech PCA Plot with 2 Principal Components before Clustering*



6.4.5. X_tech UMAP before Clustering



6.5. Use of Generative AI Tools

I used ChatGPT, Claude AI, and Gemini in Google Colab to support my M2 deliverable. ChatGPT helped me understand technical concepts like PCA, UMAP, clustering stability metrics (ARI and silhouette scores), and log-transformations for skewed variables. I also used ChatGPT to generate initial code snippets for specific tasks like setting up the clustering pipeline and creating visualization functions, which I then modified and debugged for my analysis. Claude AI was used to paraphrase my findings and generate concise, formal text for the report and code notebook explanations. Gemini in Colab assisted with debugging errors and providing quick syntax help during coding sessions.

All core analytical work was completed independently by me, including: defining the feature sets (X_tech and X_comm), making data cleaning decisions, selecting and engineering features, running clustering experiments, interpreting results, conducting sensitivity analyses, and drawing conclusions. The GenAI tools were used to enhance conceptual understanding, improve code efficiency, and refine written explanations—not to drive the substantive analytical decisions.

Link to Claude's Chat: *Claude Link*
Link to ChatGPT's Chat: *Chatgpt Link*

6.6. Special Notes

Given the page constraints for Phase M2, I used Generative AI to help streamline parts of my write-up. I might have missed some analysis due to the constraint, but please refer to *Link to Main Code File* in Appendix (6.2) to access my complete notebook.