Characterizing High-Activity Contributors in Data-Science Repositories

Overview

1. Presenting Myself
2. Research Interests Overview
3. Paper

lab-soft
software engineering laboratory

Gustavo Vale

lab·soft
software engineering laboratory

Education

B.S. Information Systems
Federal University of Lavras (UFLA)

Master in Computer Science
Federal University of Minas Gerais (UFMG)

PhD in Computer Science
Saarland University, Germany

Post-Doctorate in Computer Science
Federal University of Minas Gerais (UFMG)

Experience

Visiting Professor (2026 - Current)
UFLA, Lavras, Brazil

Co-Founder (2025 - Current)
AgroHub, Lavras, Brazil

Professor (2023 - 2025)
Unilavras, Lavras, Brazil

Experience (Cont.)

Professor (2022 - 2025)
Fagammon, Lavras, Brazil

Founder (2023 - 2025)
Grupo Vale, Lavras, Brazil

PhD Intern (2022)
Meta (ex-Facebook), London, United Kingdom

Researcher (2020 - 2024)
Universität des Saarlandes, Saarbrücken, Germany

Senior IT Consultant (2018 - 2022)
msg systems, Passau, Germany

Research Assistant (2016 - 2020)
Passau Universität, Passau, Germany

Research Assistant (2014-2016)
UFMG, Belo Horizonte, Brazil

Project Manager (2010-2013)
Comp Júnior, Lavras, Brazil

Intern (2009-2012)
Diretoria de Gestão da Tecnologia da Informação (DGTI -UFLA), Lavras, Brazil

Research Interests

lab-soft
software engineering laboratory

Research Topics

- Coordination in Software Engineering
- Artificial Intelligence (LLMs, AI agents, Predictions and Forecast Analysis)
- Software Metrics and Software Quality
- Technical Debt
- Software Analysis and Evolution
- Empirical Methods
- Human Factor in Software Engineering

lab-soft
software engineering laboratory

Research Projects

- Avaliação da Qualidade de Código Gerado por Inteligência Artificial na Resolução de Dívidas Técnicas e Conflitos de Integração em Projetos Reais
- LLM4IoT: Detecção e Correção de Falhas de Interação de Dispositivos com Grandes Modelos de Linguagem em Sistemas de Software IoT
- Avaliação da Qualidade de Código de Teste Gerado por Inteligência Artificial em Aplicações para Dispositivos Móveis

CNPq

FAPEMIG

Paper

Context

Data science is everywhere today, from healthcare to agriculture, from finance to AI research.

But behind every data science model, there is something we rarely talk about: **software**.

Context

Data science is everywhere today, from healthcare to agriculture, from finance to AI research.

But behind every data science model, there is something we rarely talk about: **software**.

Data-science projects are not just notebooks and models anymore.

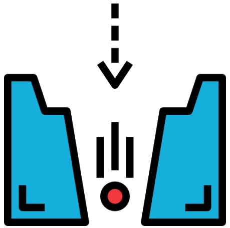
They are complex software systems combining analytics, infrastructure, and user interfaces.

lab-soft
software engineering laboratory

Opened Gap

When we look at data-science repositories on GitHub, several questions emerge:

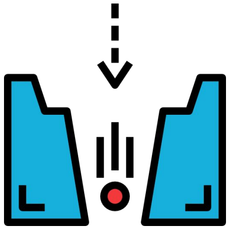
- Who actually builds these systems?
- What kinds of work do contributors perform?
- Are contributors specialists or multi-language engineers?



Opened Gap

When we look at data-science repositories on GitHub, several questions emerge:

- Who actually builds these systems?
- What kinds of work do contributors perform?
- Are contributors specialists or multi-language engineers?

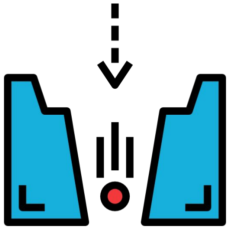


We know a lot about data-science artifacts like notebooks and pipelines, but we know much less about **the people behind them**

Opened Gap

When we look at data-science repositories on GitHub, several questions emerge:

- Who actually builds these systems?
- What kinds of work do contributors perform?
- Are contributors specialists or multi-language engineers?



We know a lot about data-science artifacts like notebooks and pipelines, but we know much less about **the people behind them**

So the question becomes: **what does contribution actually look like in data-science repositories?**

Common Knowledge

What We Know

- Data-science repositories contain notebooks, pipelines, ML models
- Python dominates
- Projects mix research and engineering

What We Don't

- How contributors distribute their effort
- How roles emerge
- How developers navigate multi-language stacks

What We Know

- Data-science repositories contain notebooks, pipelines, ML models
- Python dominates
- Projects mix research and engineering

What We Don’t

- How contributors distribute their effort
- How roles emerge
- How developers navigate multi-language stacks

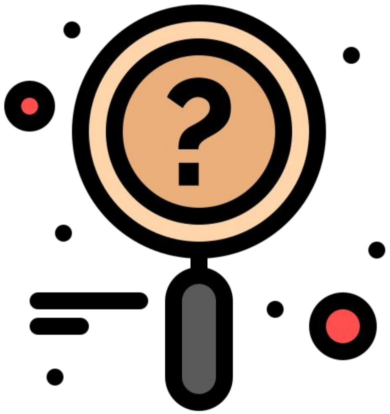
Our study tries to uncover the **structure of contribution** in data-science software

Study Overview

Study Goal

To reveal the **underlying patterns of contributor activity** and **technological scope** that shape the evolution of data-science repositories

Research Questions

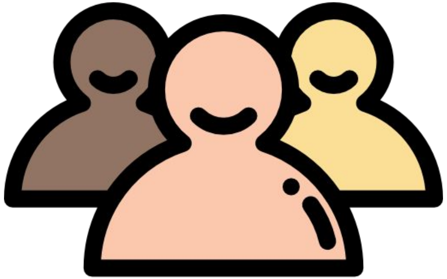


- RQ1 - Project activity patterns
- RQ2 - Dominant Programming languages in DS repos
- RQ3 - Contributor coding behaviors
- RQ4 - Mono- and Multi-language contributor profiles

Together, these questions allow us to **understand** both **projects** and **people**

Definitions

#1 - Contributor Behavior



$$r = \frac{\textit{Additions}}{\textit{Additions} + \textit{Deletions}}$$

Addition-leaning: r ≥ 0.6 → feature development

Balanced: 0.6 > r > 0.4 → iterative development

Deletion-leaning: r ≤ 0.4 → refactoring & cleanup

#2 - Programming-language Mapping

- **Core data-science Languages**: py → Python, ipynb → Jupyter, r → R
- **Web/front-end**: js, jsx → JavaScript; ts, tsx → TypeScript; html → HTML
- **Systems and back-end**: c → C; cpp, cc → C++; java → Java
- **Configuration and infrastructure**: yaml → YAML; json, jsonc, jsonl → JSON
- **Documentation and text**: md → Markdown; tex → LaTeX; txt → Text
- **Data and assets**: csv, tsv → CSV; binary archives (e.g., zip, tar, gz, tgz, whl) are mapped to generic binary/data categories

Code vs. non-code classification

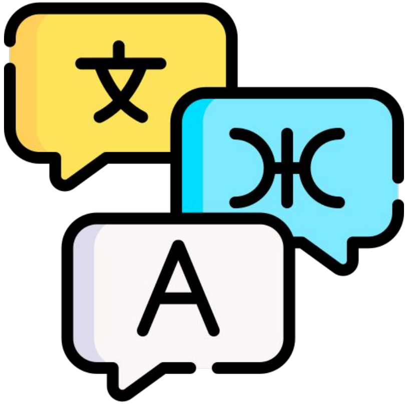
#3 - Mono- and Multi-language Contributors

NLC = | *Ld* |

NLC stands for Number of Language related to Code

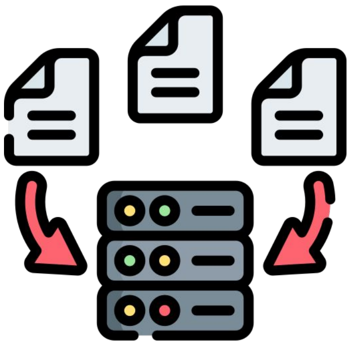
For each contributor – project pair, let *Ld* be the set of distinct code languages in which the contributor has modified at least one file;

- **No-code contributors**: *NLC* = 0
- **Mono-language contributors**: *NLC* = 1
- **Multi-language contributors**: *NLC* ≥ 2



Method

Data Collection



Repositories:

- ML frameworks
- Pipelines
- Monitoring tools
- Visualization tools

Data extracted from Git with PyDriller

Selection criteria:

- ≥100 stars
- ≥5 contributors
- active development

Dataset

- 18 GitHub repositories
- 65 high-activity contributors

Repository	Domain	Stars	Contributors	Forks
Aliro	Bioinformatics	219	20	63
BastionLab	Secure Computation	165	12	11
Colour	Color Science	1.9k	45	266
DeepVariant	Genomics	3.1k	24	741
Gop	DSL Compiler	8.8k	39	549
Kedro	Data Pipelines	9.3k	211	936
Lale	AutoML Framework	320	25	81
LineaPy	Experiment Management	653	21	57
Metaflow	Workflow Management	7.5k	88	824
MLOS	ML Optimization	123	18	71
NannyML	ML Monitoring	1.7k	29	153
Nebari	Infrastructure	254	63	98
PySyft	Federated Learning	9.2k	423	2.0k
Python-AIPlatform	Cloud AI SDK	520	93	360
Quadratic	Data Visualization	2.7k	22	195
SystemDS	Distributed Analytics	1.0k	180	482
VerticaPy	Analytics API	214	16	47
VisualPython	Data Analysis Tools	799	6	115

DSL stands for Domain Specific Language

Metrics Dimensions

Activity

- commits
- modified files
- code churn

Behavior

- additions vs deletions

Technology

- programming languages per contributor

We combine simple metrics with unsupervised learning (PCA + clustering)

Results

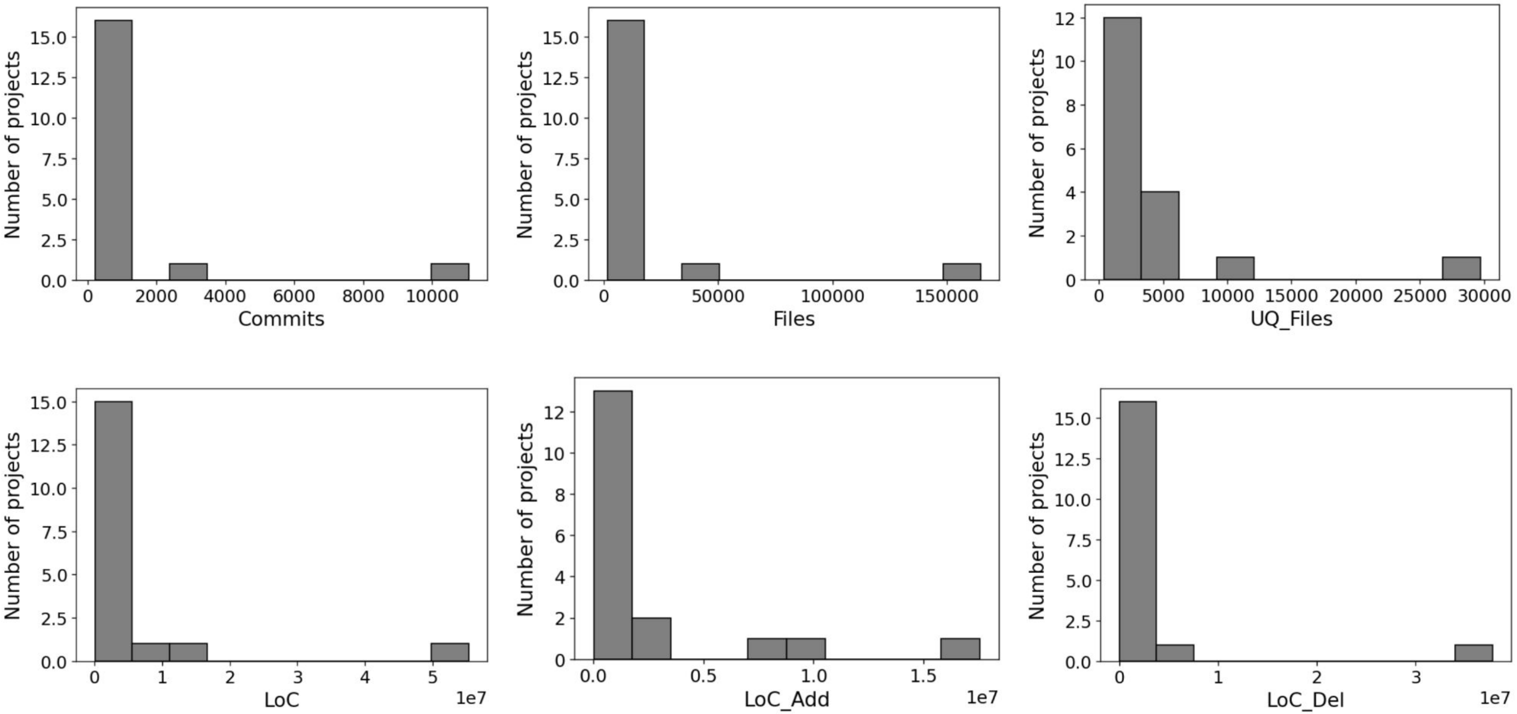
RQ1 – Project-level Activity Patterns

lab-soft
software engineering laboratory

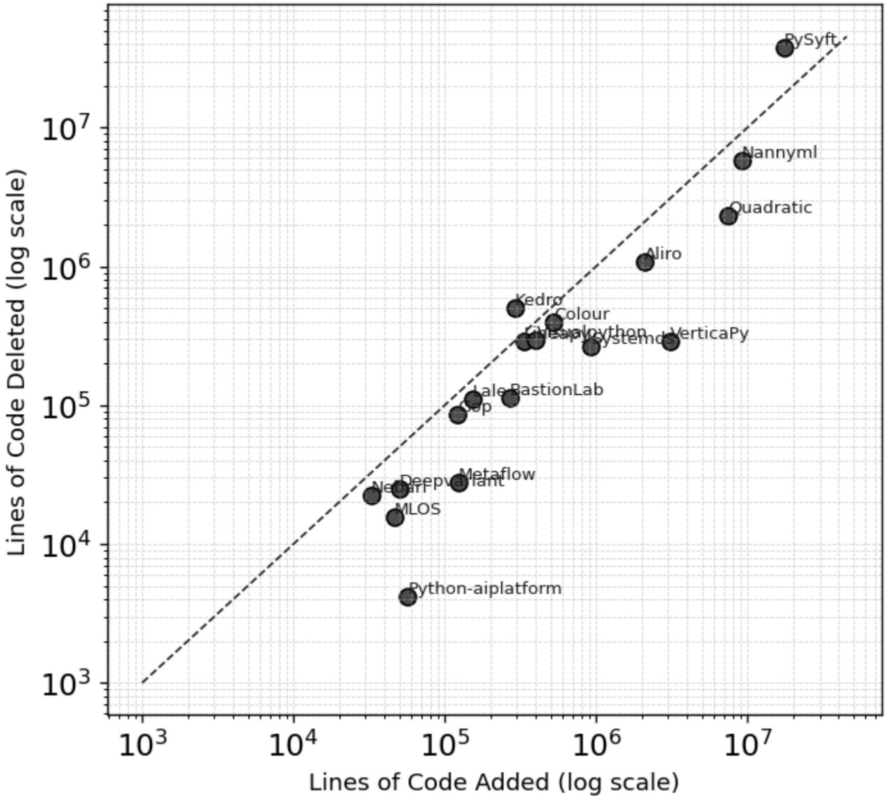
Descriptive Statistics

Project	Commits	Files	UQ_Files	LoC	LoC_Add	LoC_Del
Aliro	668	12,192	4,252	3,190,163	2,105,122	1,085,041
BastionLab	955	5,222	1,271	380,252	267,249	113,003
Colour	831	15,959	1,573	918,174	522,918	395,256
DeepVariant	419	1,918	789	74,950	50,089	24,861
Gop	1,171	4,468	804	206,907	121,764	85,143
Kedro	514	12,785	3,539	789,506	289,427	500,079
Lale	337	1,535	617	263,721	153,700	110,021
Lineapy	919	8,953	3,792	618,755	332,496	286,259
MLOS	302	3,000	376	62,220	46,593	15,627
Metaflow	485	1,918	1,531	94,611	122,416	27,805
NannyML	856	7,646	2,164	15,009,836	9,163,316	5,846,520
Nebari	213	2,133	1,532	55,068	32,671	22,397
PySyft	11,065	164,746	29,711	55,338,698	17,535,927	37,802,771
Python-aiplatform	342	2,071	959	61,268	57,072	4,196
Quadratic	2,977	37,960	9,786	9,733,895	7,408,439	2,325,456
SystemDS	569	9,496	4,450	1,192,345	925,940	266,405
VerticaPy	490	7,676	1,796	3,371,456	3,085,162	286,294
VisualPython	844	9,044	3,210	695,353	401,692	293,661

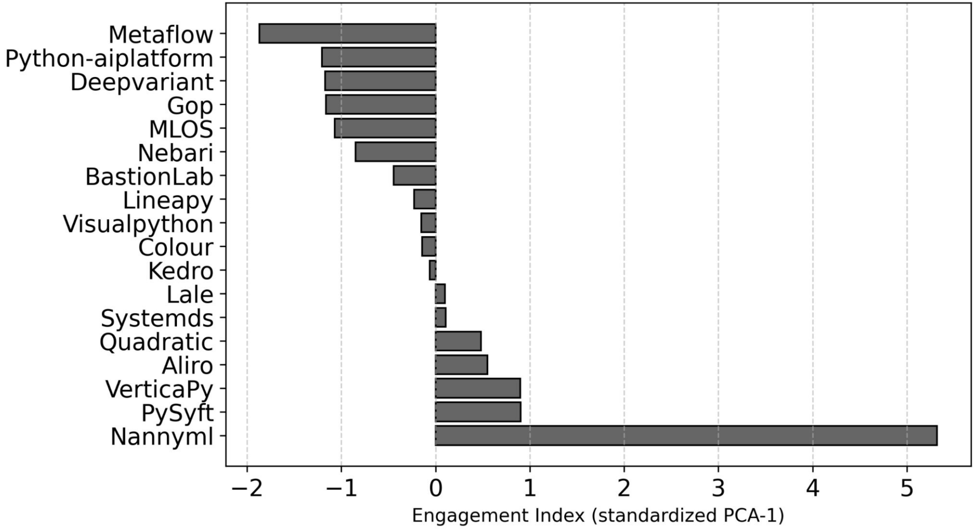
Descriptive Statistics

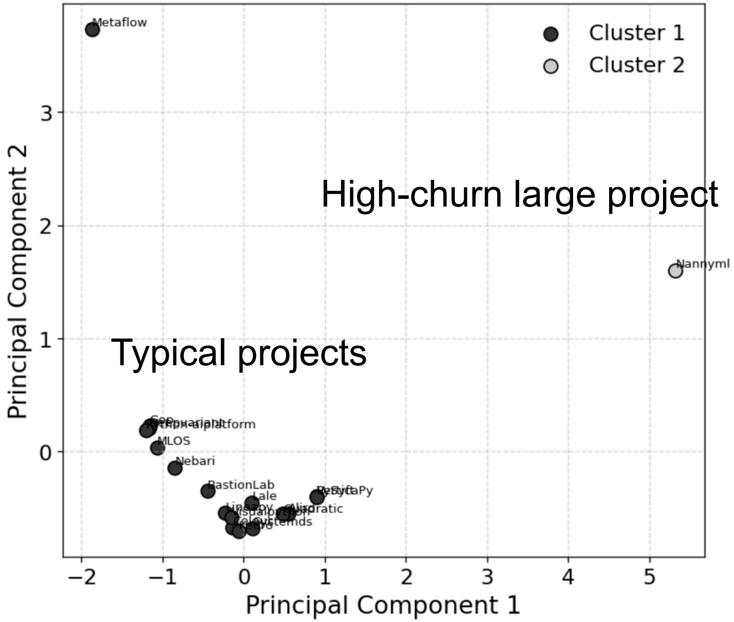


Code Growth vs. Deletion



Normalized Engagement and Archetypes





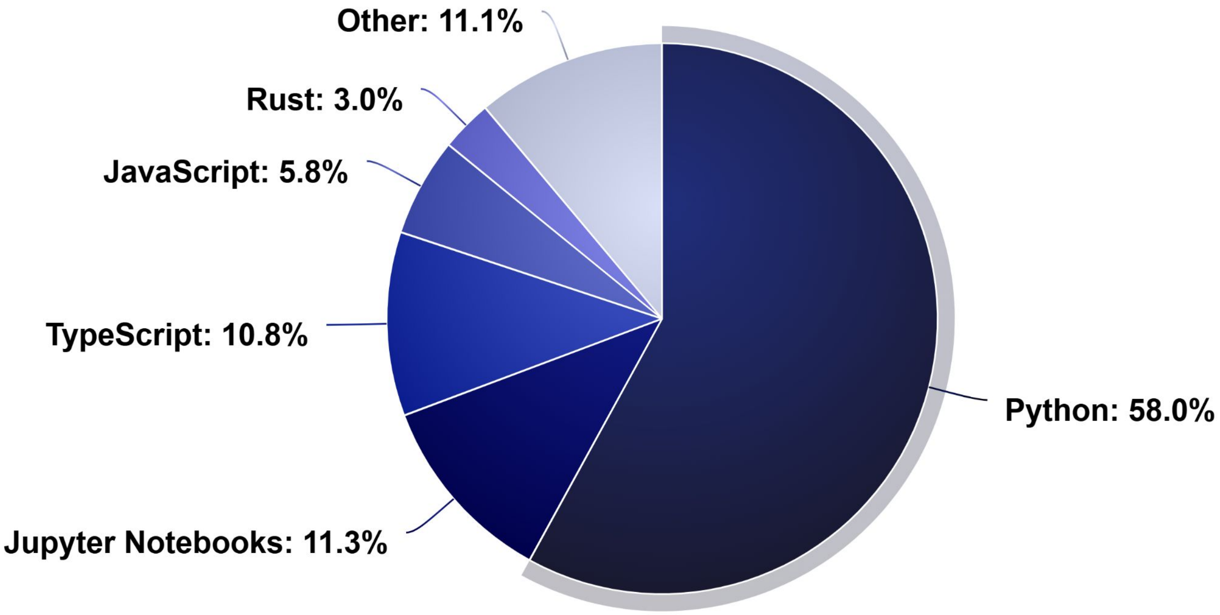
Summary RQ1 - Project-level Activity Patterns

Activity in open-source data-science projects **is intense but unevenly distributed** even when normalizing the codebase

Project-level engagement follows a long-tail pattern, with diverse evolution strategies, ranging from incremental growth to aggressive refactoring, coexisting within the data-science OSS ecosystem

RQ2 – Dominant Programming Languages and Project Composition

Global Prevalence of Programming Languages



6. HTML (2.7%),
7. Java (2.1%),
8. Svelte (1.8%),
9. Go (1.6%),
10. Shell (1.2%)

Other (1.7%)

Code-only Language Composition per Project

Python is the primary implementation language in 13 of 18 repositories

JavaScript for `Aliro` and `VisualPython` repositories

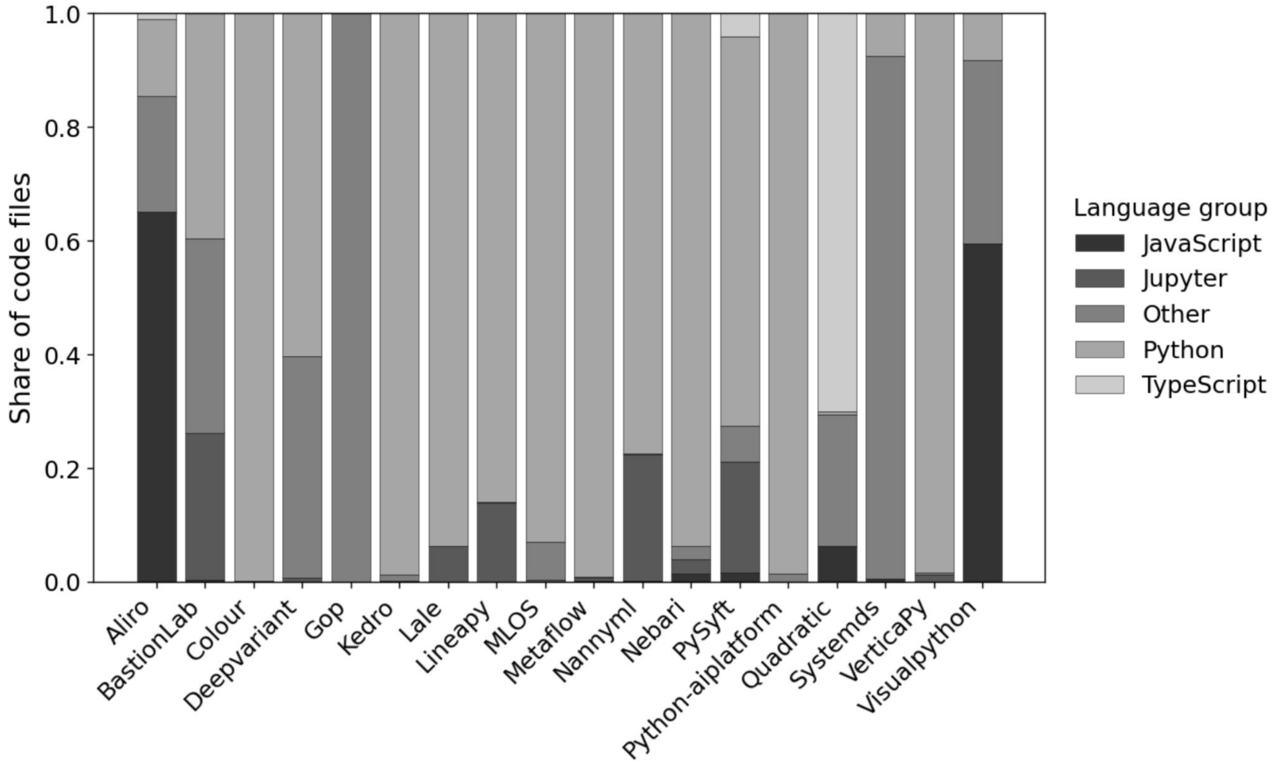
TypeScript for `Quadratic`

Go for `Gop`

Java for `SystemsDS`

Jupyter notebooks play an important but secondary role (share ≥10%) in four projects: BastionLab (25.7%), NannyML (22.2%), PySyft (19.4%), and Lineapy (13.8%).

Stacked project-level Language Composition



Summary RQ2 - Dominant Programming Languages

Repositories are overwhelmingly Python-centric

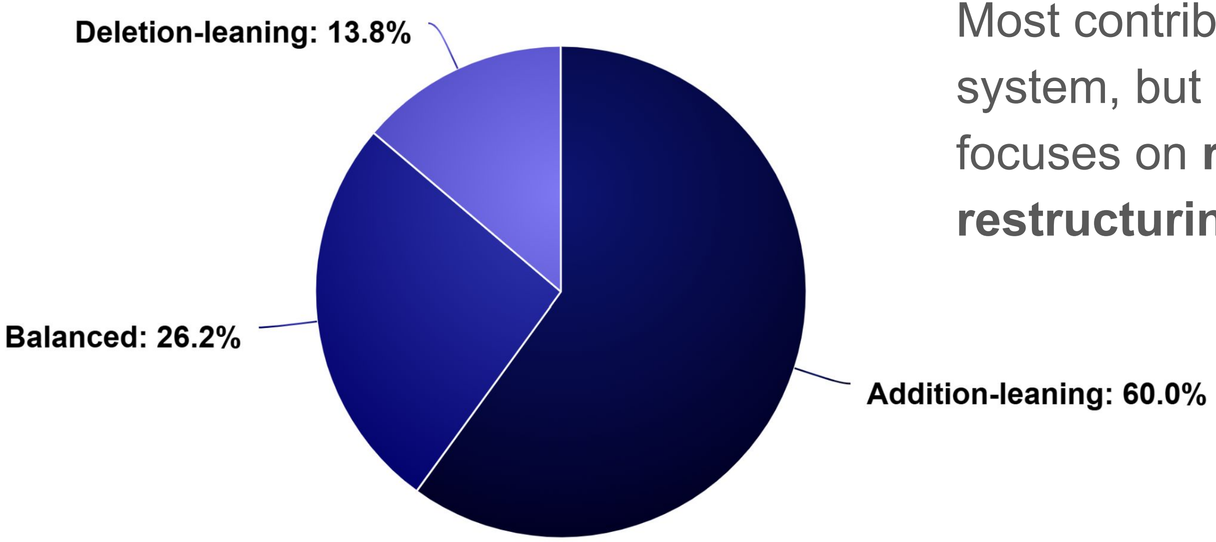
Python accounts for 58% of all code files globally and is the primary language in more than two-thirds of the projects

A layered composition in which:

- **Python** libraries form the **core analytic** and **orchestration logic**
- **Notebooks** support **experimentation**
- **Web-oriented stacks** provide **user-facing interfaces** and **supporting infrastructures**

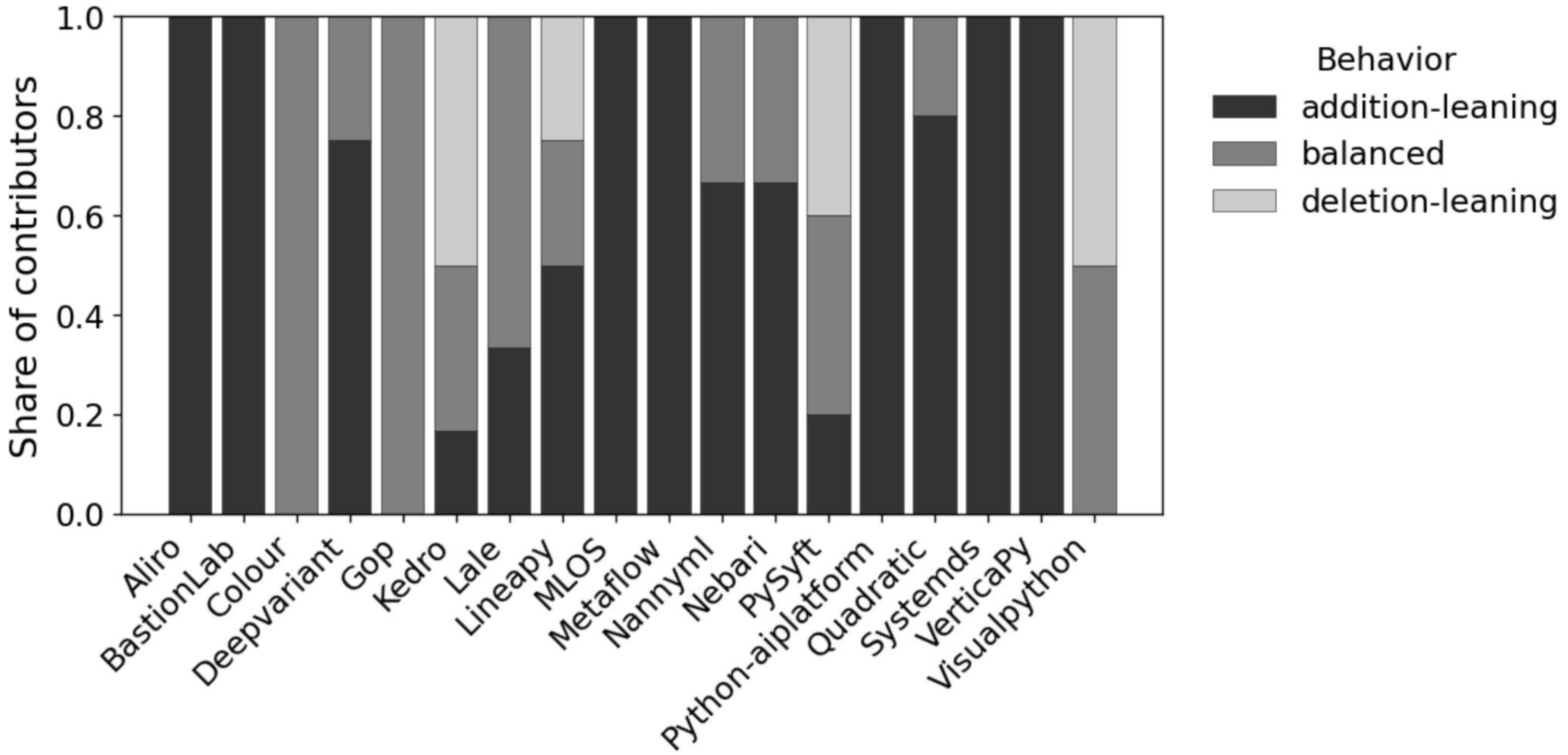
RQ3 – Coding Behaviors of Contributors

Behavior Categories and Prevalence

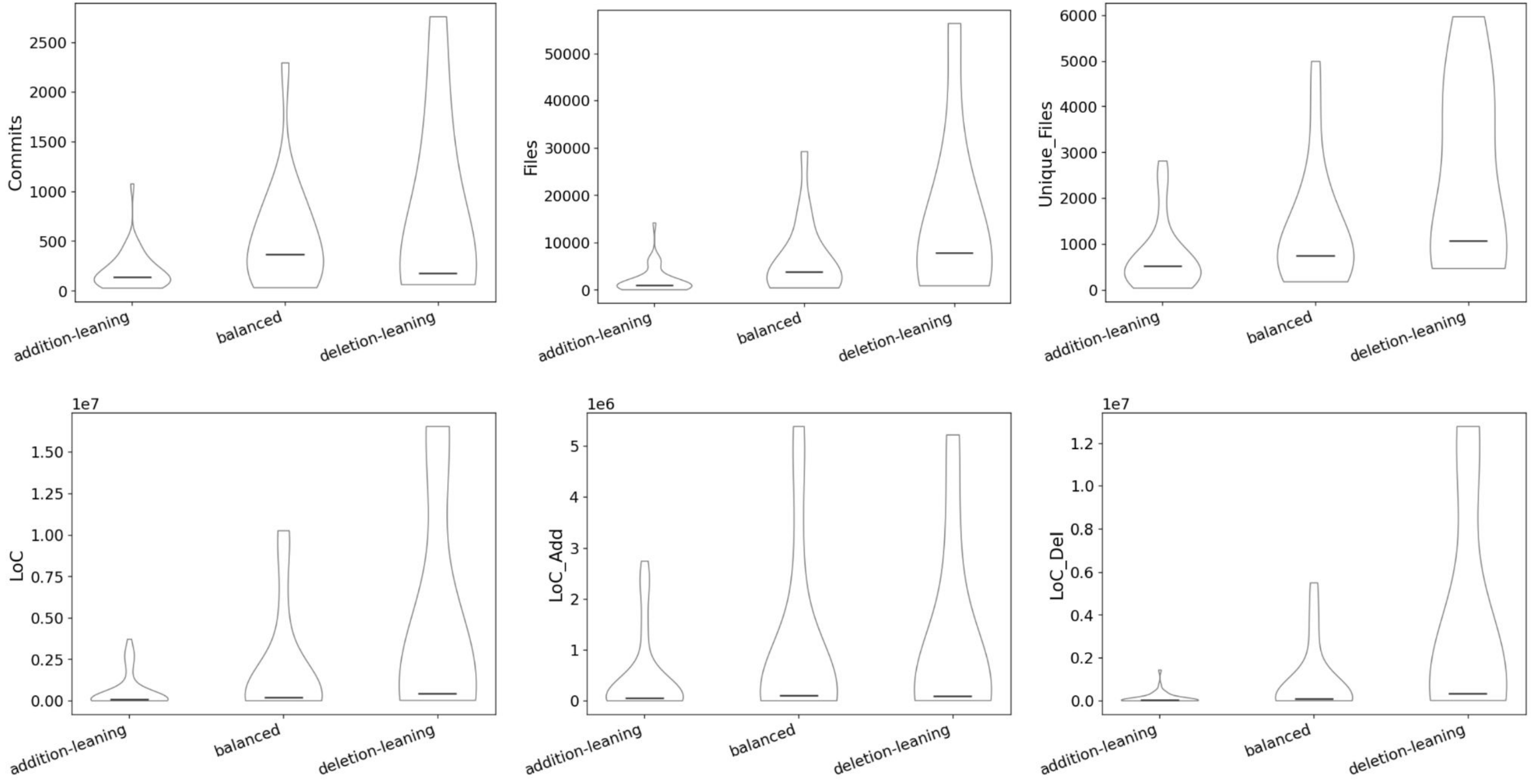


Most contributors extend the system, but a smaller group focuses on **removing** and **restructuring code**

Behavioral Variation Across Projects

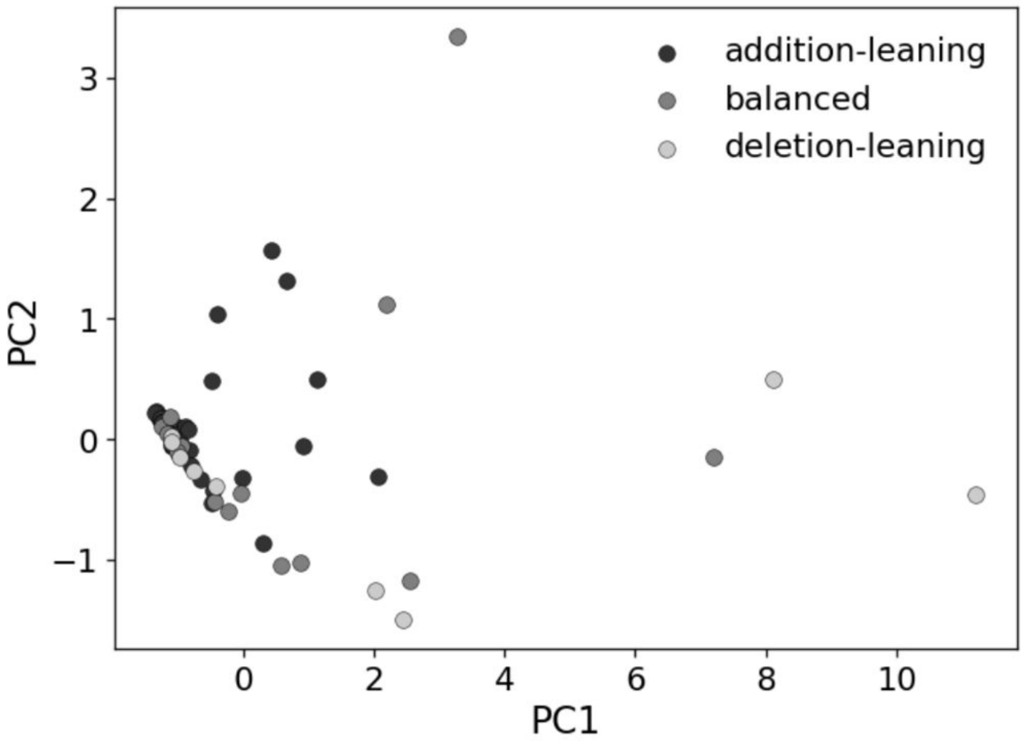


Activity Distributions by Behavior



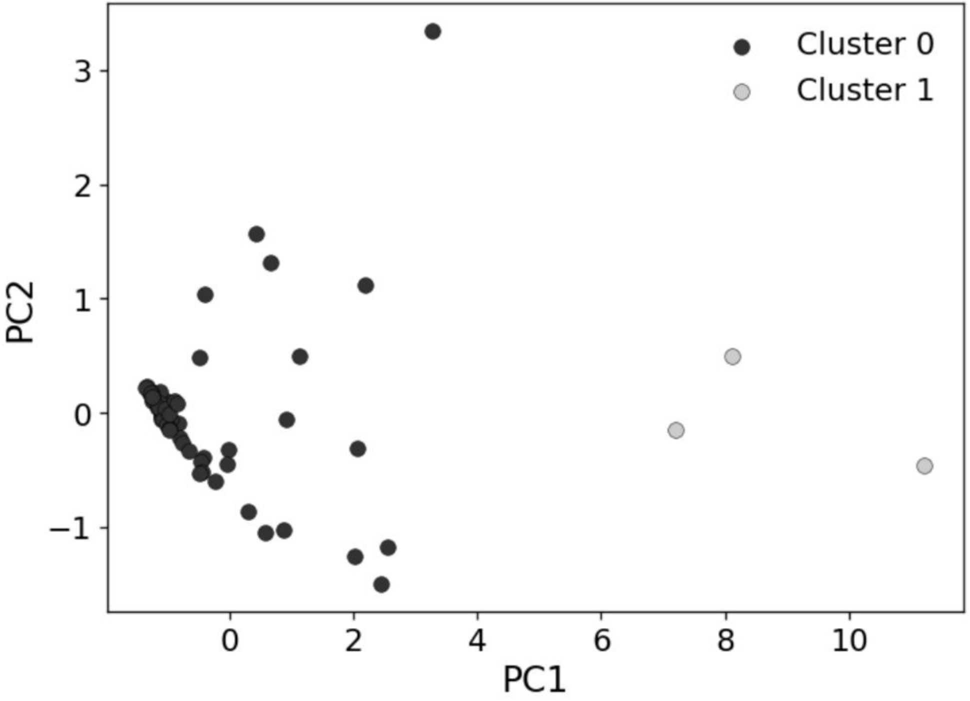
PCA View of Contributor Activity

Represents 92.8% of the variance



Unsupervised Clusters of Contributor Profiles

Represents 92.8% of the variance



Summary RQ3 - Coding Behaviors of Contributors

A **majority of contributors are addition-leaning**

A **sizeable group of balanced** contributors perform both additions and deletions at relatively high intensity;

A **smaller subset of deletion-leaning** maintainers carry out large-scale removals and refactorings across many files.

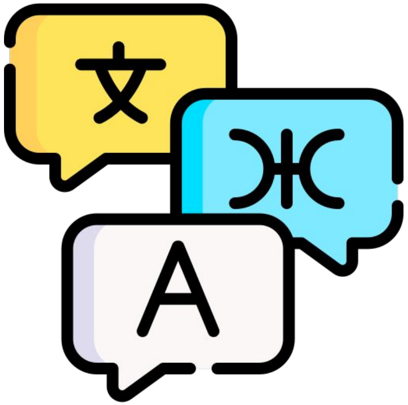
PCA shows that most variance in contributor activity is driven by overall churn volume, while unsupervised clustering (with k = 2) separates routine contributors from a small group of high-intensity, deletion-heavy maintainers.

Together, these results reveal a heterogeneous division of labor in the evolution of data-science projects, with distinct roles for growth-oriented contributors, balanced contributors, and clean-up oriented maintainers

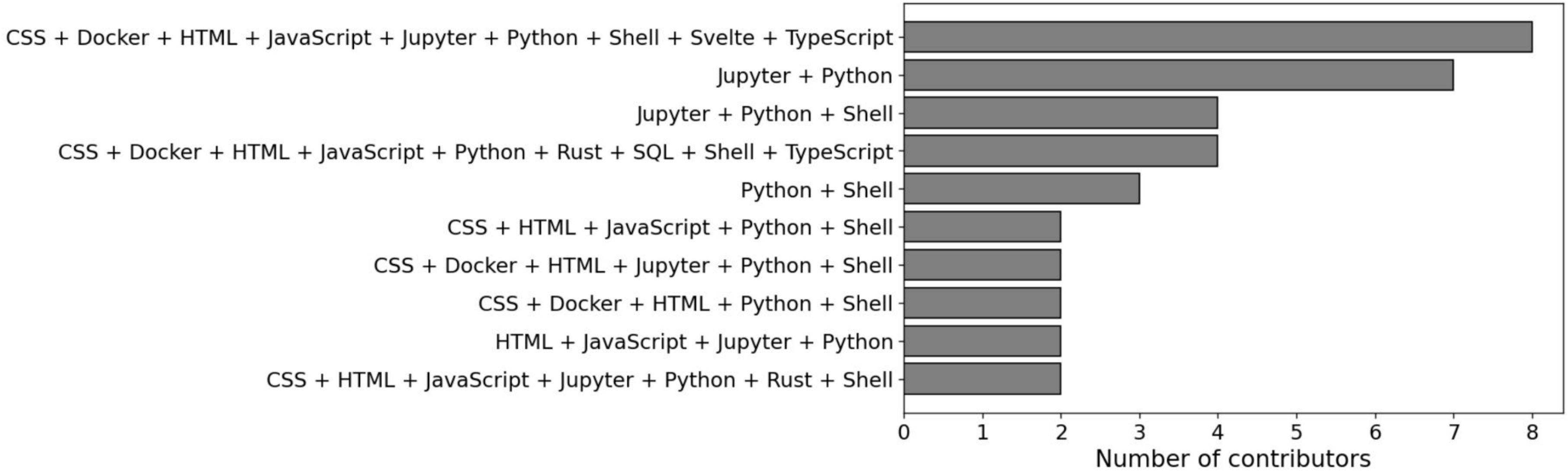
RQ4 – Multi-language Contributor Profiles

Mono- vs. Multi-language Participation

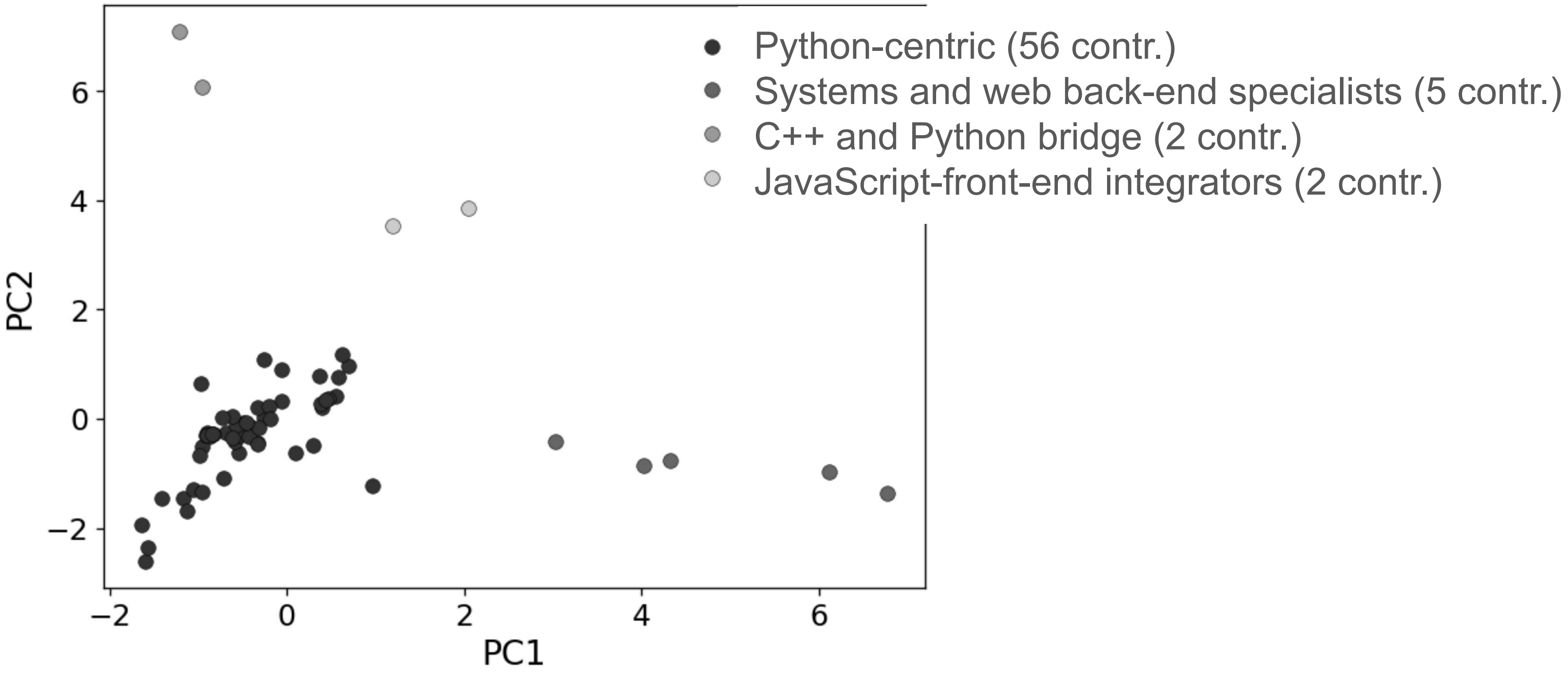
- 6 contributors (9%) remained mono-language (4 uses only Python)
- 59 edited code in at least 2 programming languages
- The number of programming varied from 1 to 9 (average 5.4 languages)
- 30% of 59 use 8 or more languages



Top Multi-language Bundles



PCA and Clustering on Language Profiles



Linking Unsupervised Clusters to Mono-/multi-language

Cluster	Mono-language	Multi-language
0	6	50
1	0	5
2	0	2
3	0	2

Summary RQ4 – Multi-language Contributor Profiles

Multi-language participation is the norm using around 5 programming languages

Python and Jupyter form the core of most contributors' portfolios and often co-occur with web technologies, scripting languages, and infrastructure tools

Data-science ecosystems rely on a highly multi-language workforce, where **contributors assume distinct technological roles** rather than simply adding more languages to a flat skill set

Discussion

Rethinking Roles in Data-Science Repositories

Industry and research often separate “**data scientists**” and “**software engineers**”, sometimes adding “**research software engineers**”

Our findings suggest that data-science roles are better described by **behavioral and technological profiles** (activity patterns and language portfolios) than as static job titles

Coarse labels risk overlooking contributors to the sustainability and productionization of data-science systems

The Importance of Deletion-Leaning Contributors

A small and critical group specialized in deletions (e.g., refactoring and removing obsolete code) **is often undervalued**

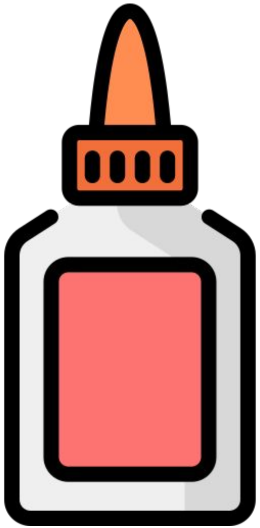
The concentration of these critical "code curators" raises concerns about project resilience

We advocate for the **systematic recognition, support, and distribution of maintenance and refactoring responsibilities**, rather than relying on a few individuals

Multi-Language Work as “Glue” Across Layers

Multi-language contributors act as a form of “glue” that connects different layers of data-science systems

Multi-language contributions are central to the structural integrity of data science software, not just side effects of heterogeneous tool-chains.



Project-Level Engagement Archetypes and Governance

Data science **repositories are diverse**

- Highly active projects with broad participation benefit from strict guidelines, while moderately active projects rely on a small core for maintenance and refactoring

These differences influence governance

- Project stage and scope also affect the balance between exploratory work (prioritized by younger projects) and stabilization/sustainability (focused on by mature projects, often with deletion-leaning and multi-language contributors)

Understanding a **project's engagement archetype** is essential for guiding process, onboarding, and resource allocation decisions

Implications for Practitioners, Researchers, and Tool Builders

Early-career contributors: Our results highlight multiple viable pathways into data-science repositories

Experienced practitioners and project maintainers: Our analysis suggests the importance of making deletion-leaning and multi-language work visible and valued

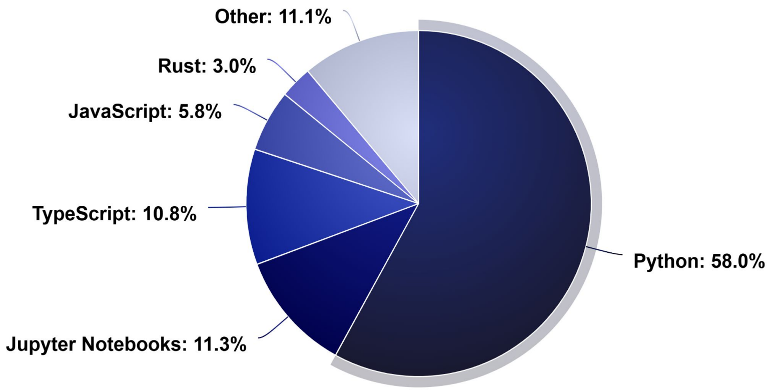
Researchers: Combining simple ratio-based metrics with unsupervised analyzes can reveal nuanced contributor profiles in data-science settings

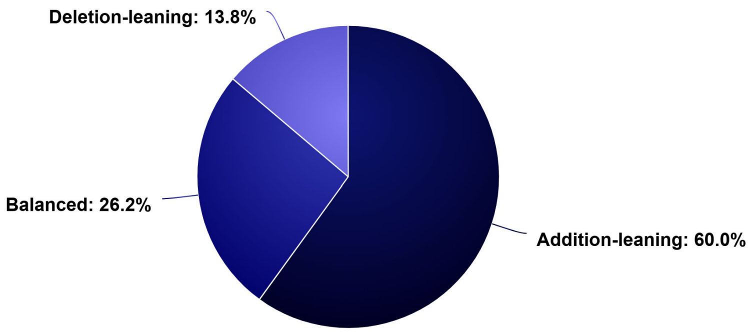
Tool builders: Our results motivate role-aware and language-aware analytics. Development environments and project dashboards can highlight behavioral categories to improve coordination, ensure fairer recognition, and promote sustainable evolution in data science projects

Conclusion

Who are the contributors?

RQ1 - Data-science repositories exhibit heterogeneous engagement archetypes





lab-soft
software engineering laboratory

UFMG

gustavovale@dcc.ufmg.br

@gustavovaleoficial