Highlights

Not Just for Developers: How Knowledge-Centric Communities Build and Sustain Plugin Ecosystems

Giovanni Rosa, David Moreno-Lumbreras, Raula Gaikovina Kula

- LLM-based topic modeling on all 2,675 Obsidian community plugins identifies 60 topics organized into six thematic families spanning knowledge management, AI assistance, and editor customization.
- The ecosystem exhibits a modular, long-tailed structure with semantic proximity among related leaf topics and representation-space separation among the six higher-level families.
- Closed pull requests and issues consistently exceed open ones across the topic families; these states describe repository activity but do not identify contributor demographics or the outcome quality of closed items.
- 60.9% of the 2,724 mined repositories contain a GitHub Actions workflow, used here as a repository-automation proxy; workflow presence is only weakly associated with stars ($\rho = 0.10$).
- An emerging AI-powered plugin cluster blurs the boundary between coding and thinking, signaling that knowledge-centric communities are a new frontier for software ecosystem research.

Not Just for Developers: How Knowledge-Centric Communities Build and Sustain Plugin Ecosystems

Giovanni Rosa[a], David Moreno-Lumbreras[a,*] and Raula Gaikovina Kula[b]

[a]Universidad Rey Juan Carlos, Spain
[b]The University of Osaka, Japan

ARTICLE INFO

Keywords:
Software ecosystems, software maintenance, Knowledge-centric communities.

ABSTRACT

Third-party extensions underpin large software ecosystems, but empirical research has primarily examined package and plugin platforms oriented toward software-production tasks. Less is known about extension ecosystems whose user-facing functionality supports knowledge work. In this paper, we study *Obsidian*, a knowledge-centric platform whose community—focused on writing, organization, and creativity—has built a substantial plugin ecosystem around those workflows.

We investigate what kinds of plugins exist within this hybrid ecosystem, how the ecosystem is structurally organized, how plugins are maintained and evolved, and what development practices plugin developers adopt. Using repository mining and LLM-based topic modeling on the complete ecosystem of 2,675 community plugins, we identify 60 leaf topics organized into six broad thematic families covering knowledge management and tooling domains, from task and productivity management to AI-powered writing assistance and editor customization.

Structural analysis reveals a modular, long-tailed ecosystem with semantic proximity among related leaf topics and representation-space separation among the six higher-level families.

Furthermore, analysis of pull requests, issues, commits, and repository automation documents several complementary dimensions of repository activity without treating any one of them as a direct measure of maintenance quality. Of the 2,724 unique repositories mined, 1,660 (60.9%) contain at least one GitHub Actions workflow; among the 1,068 repositories with recorded runs, the median observed success proportion is 0.95. Workflow presence has only a weak association with stars ($\rho = 0.10$) and a negligible association with repository age ($\rho = -0.05$).

Our findings characterize the functional structure, recorded repository activity, and automation practices of this ecosystem, while identifying the additional evidence required to assess maintenance quality and long-term sustainability.

1. Introduction

Many researchers and practitioners are likely to assume that software ecosystems are environments created *by developers and for developers*. Platforms such as Visual Studio Code, NPM, PyPI, and JetBrains IDEs represent this classical view: ecosystems in which plugins and extensions support programming workflows, accelerate development, enable automation, and integrate specialized engineering tools [8]. Within this paradigm, ecosystem evolution, maintainability, and governance have been studied primarily through the lens of software engineering practices, with the implicit assumption that both producers and consumers share technical expertise.

However, modern software ecosystems increasingly support large and diverse communities whose goals extend far beyond programming.

In these environments, plugins extend activities such as writing, learning, planning, and organizing information rather than only programming workflows. This shift raises questions about how extension ecosystems form and evolve when their application domain is knowledge work; it does not, by itself, identify the technical background of either users or contributors.

Another motivation is how there is need for a richer understanding of hybrid software ecosystems that extend beyond developer-centric ecosystems, especially in the era of Generative Artificial Intelligence (GenAI). As large language models further reduce the barriers to customization and automation [2, 18], the boundary between coding

*Corresponding author
giovanni.rosa@urjc.es (G. Rosa); david.morenolu@urjc.es (D. Moreno-Lumbreras); raula-k@ist.osaka-u.ac.jp (R.G. Kula)
ORCID(s): 0000-0002-5241-1608 (G. Rosa); 0000-0002-5454-7808 (D. Moreno-Lumbreras); 0000-0003-2324-0608 (R.G. Kula)

tasks and thinking tasks becomes increasingly blurred. Platforms like Obsidian illustrate this transition: developers provide plugins that enable users to construct, navigate, and reason about knowledge through software artifacts.

Obsidian—a Markdown-based knowledge management application—is a compelling example of this emerging class of hybrid software ecosystems. Originally designed as a note-taking environment, Obsidian has grown into a modular, community-driven platform enhanced by thousands of plugins hosted on GitHub. These plugins span a wide spectrum of functionality: from task and calendar management, to AI-powered writing assistants, to visualization tools, diagram editors, music notation utilities, and flashcard generation. Crucially, these plugins do not extend a programming environment. They extend a set of *knowledge-centric workflows*: writing, structuring information, planning tasks, externalizing ideas, and building personal knowledge systems.

This combination of software production and knowledge-work functionality poses questions for software engineering research. What kinds of software artifacts emerge in such hybrid ecosystems?

How are functions for writing, planning, and reasoning organized into plugins? What repository activity and development practices are observable around those plugins? A matched cross-ecosystem comparison would be needed to establish resemblance to or divergence from developer-centric platforms.

Despite its rapid adoption and vibrant plugin ecosystem, Obsidian has received *limited systematic attention* from the software engineering community, especially when compared to ecosystems such as VS Code or the Chrome Web Store [26]. This absence of formal analysis obscures fundamental questions: how the ecosystem is organized, which functionalities dominate, how specialized its extensions are, or how developers maintain them over time. Moreover, Obsidian sits at the intersection of software engineering, human–computer interaction, personal knowledge management, and cognitive augmentation—making it an ideal setting to explore emerging hybrid software ecosystems.

Motivated by these gaps, this paper presents an empirical study of the Obsidian plugin ecosystem from a software engineering standpoint. This paper presents an extension of our previous work [27], which introduced a preliminary taxonomy and maintenance analysis of a subset of Obsidian plugins using two research questions (RQ1 and RQ3).

The present paper extends that work in four key ways: (i) we analyze the complete ecosystem of 2,675 community plugins, compared to the subset used in the short paper; (ii) we add two new research questions—RQ2 (ecosystem structure analysis) and RQ4 (development practices including GitHub Actions workflow presence and activity, correlation analysis, and cross-topic comparisons)—that were not present in the original study; and (iii) we substantially expand the methodology with a refined *BERTopic* pipeline, hierarchical topic grouping, and repository-level metrics mining; and (iv) we conduct a stratified manual validation of LLM-generated keyphrases, leaf-topic assignments, and parent-family assignments, including independent annotation and third-annotator adjudication.

We combine repository mining, clustering, and large language models (LLMs) to uncover the functional landscape of the ecosystem and to analyze the extent to which plugins are maintained. By analyzing the complete ecosystem of 2,675 community plugins, we provide a comprehensive characterization of how a knowledge-centric community organizes, develops, and sustains its software extensions.

Our investigation is guided by the following research questions:

- ***RQ1: What kinds of plugins exist within the Obsidian ecosystem?*** This question is motivated by the need to understand how functionality is organized in a hybrid ecosystem where developers build plugins for a knowledge-centric community, rather than for programming workflows.
- ***RQ2: How is the Obsidian plugin ecosystem structurally organized?*** Beyond identifying individual topics, we characterize the inter-topic relationships within Obsidian: which functional areas are closely related and which are semantically isolated. References to other ecosystems are contextual rather than a matched quantitative comparison.
- ***RQ3: To what extent are these plugins actively maintained and evolved?*** Because the repositories support a knowledge-management platform, we examine their maintenance traces without assuming the demographics of either plugin users or contributors and without making a direct cross-ecosystem comparison. We analyze pull requests, issues, and commits as indicators of ecosystem maintenance.
- ***RQ4: What development practices characterize the Obsidian plugin ecosystem?*** Beyond maintenance activity, we examine repository practices including GitHub Actions workflow presence and recorded run activity, ecosystem growth trajectory, and the relationships between development metrics such as stars, commits, issues, and workflow activity.

Our results characterize how repositories supporting a knowledge-centric platform evolve; they do not identify the demographics or professional background of their contributors. We discuss the implications of these findings for the broader software engineering community. A complete reproduction package, including all scripts, processed data, and artifacts used in this study, is publicly available for inspection and reuse [28].

2. Background and Related Work

2.1. Software Ecosystems and Knowledge-Centric Communities

Software ecosystems are networks of actors that co-create value around a shared technological platform, including through third-party extensions [4, 12, 32]. Plugin ecosystems are a platform-dependent form of this model: extensions add user-facing capabilities but must also evolve with the host platform and its interfaces. Prior work therefore emphasizes contribution dynamics, coordination, and community structure in open-source sustainability [6, 21], as well as co-evolution and compatibility between plugins and their host platform [15].

In this study, a *knowledge-centric ecosystem* means a software ecosystem whose user-facing extensions primarily support the creation, organization, connection, retrieval, or transformation of knowledge artifacts—for example, notes, Markdown files, links, annotations, tasks, and visual representations. A *knowledge-centric community* refers to the actors and activities organized around those workflows. The adjective classifies the platform's application domain and plugin functionality; it does not assert that every user is a non-developer or that plugin contributors lack professional software-engineering experience. We use *hybrid* to distinguish the two layers: producing a plugin is a software-development activity, whereas the plugin's principal use may be writing, learning, planning, or personal knowledge management rather than software production. Accordingly, the platform's user population and its plugin-contributor population are conceptually distinct, and our repository data characterize the latter's observable artifacts rather than either population's demographics.

2.2. Plugin Lifecycle and the Obsidian Study Boundary

For this study, the plugin lifecycle comprises four stages: implementation and versioning in a source repository; inclusion and discovery through the community-plugin registry; installation and updating by users; and post-release evolution. The collected registry and linked GitHub data expose the first, second, and selected traces of the fourth stage—commits, issues, pull requests, and repository automation. We did not analyze release cadence, and the data do not expose local installation counts or usage, the substance of registry-review deliberations, user satisfaction, private development, or plugins distributed outside the registry. Thus, registry inclusion provides a reproducible boundary for the studied population, not a quality certification or a complete census of every Obsidian extension. The replication package preserves this boundary and the collection cutoff [28].

Obsidian's local-first, Markdown-based architecture makes files, links, and note transformations central extension points. Its community registry makes plugins discoverable through a curated list, while the linked GitHub repositories make commits, issues, pull requests, and GitHub Actions observable. These characteristics enable the present repository-mining design but also shape both the functional taxonomy and the available maintenance proxies. They are revisited as limits to transferability in Section 6.3.

2.3. Related Empirical Work and Positioning

Research on plugin and extension ecosystems has examined security and governance risks in developer tools and browser marketplaces [8, 11], plugin–platform incompatibilities in WordPress [15], and classification and recommendation in the Chrome Web Store [26]. Repository-mining research has also addressed application-domain classification [29, 33] and GitHub Actions categorization [24]. A complementary end-user-development literature studies how people adapt computational artifacts and the barriers encountered outside conventional programming practice [13, 14, 22]. More recent work considers human–AI collaboration and the trust, usability, and evaluation challenges of LLM-enabled tools [1, 2, 7, 23, 30].

Our preliminary study introduced a taxonomy and repository-level maintenance analysis for a subset of Obsidian plugins [27]. The present study differs by analyzing the complete registry snapshot, adding hierarchical structure and repository-practice questions, and manually validating the final LLM-assisted outputs. Its contribution is an in-depth, reproducible characterization of one knowledge-centric plugin ecosystem. It does not estimate causal effects or perform a matched quantitative comparison with NPM, VS Code, Chrome, or another ecosystem; references to those ecosystems provide theoretical context and motivate future comparative work.

3. Methodology

This section reports the experimental procedure used to construct and validate the data-driven taxonomy of the Obsidian plugin ecosystem. It is composed of six main stages: (i) Data Collection and Preparation, (ii) Topic Modeling and Labeling, (iii) Manual Validation, (iv) Ecosystem Structure Analysis, (v) Source Repositories Mining, and (vi) Development Practices Analysis.

All steps rely on accessible open-weight models to ensure future reproducibility.

3.1. Data Collection and Preparation

This study targets the complete community-maintained ecosystem of Obsidian plugins. We retrieved the full registry[1] on February 10, 2026, obtaining 2,725 plugins with metadata such as name, author, description, and repository URL. For each plugin, we fetched the corresponding README file from its GitHub repository.

Because many plugin descriptions were too short to capture functionality, we enriched them by extracting additional key phrases from each plugin's README file. Each README was processed with the open-weight LLM *gpt-oss-120b*[2] (F16 GGUF, *num_ctx*=8192, served via *Ollama*), which was prompted to generate two to four concise key phrases summarizing the plugin's usage context and functionality. The prompt, reported in the replication package [28], focused on three dimensions: (i) the problem the plugin aims to solve, (ii) its workflow integration, and (iii) the user context.

Additionally, the LLM is prompted to return `NULL` as a response, instead of the requested key phrases, when (i) the provided information is insufficient to extract meaningful key phrases or (ii) the provided content is not in English. The plugins falling into one of those cases were then removed. For some instances, we noted that the extracted key phrases were too long or simply not key phrases at all (*i.e.,* the model hallucinates generating long textual responses), exceeding the 99th percentile in terms of character length distribution. The resulting instances were treated as outliers and removed from the final dataset. After this filtering, the final dataset contained 2,675 valid plugin instances used for subsequent analysis.

3.2. Topic Modeling and Labeling

The goal of the first RQ is to identify the main themes among the analyzed plugins. To achieve this, we performed a topic modeling analysis using the *BERTopic* library[3], a widely used framework that implements the state-of-the-art topic modeling pipeline using the major NLP algorithms and tools.

The first step was to create a set of input documents. For each plugin, we produced one document by concatenating the plugin name, short description, and the key phrases extracted from the source README. The first three pieces of information are already given by the plugin's author as metadata.

The next step was to extract vector embeddings in order to analyze the encapsulated semantics of the input documents and proceed with the analysis. This was achieved using the *Qwen/Qwen3-Embedding-0.6B*[4] embedding model (F16 GGUF export, served through Ollama). The resulting embeddings were L2 normalized to enhance the measurement of semantic similarity.[5]

We applied the default *BERTopic* pipeline, taking as input the normalized embeddings, with the following steps.

1. **Dimensionality reduction**: We applied the *UMAP* algorithm [20] with the parameters *n_neighbors*=10, *n_components*=2, *min_dist*=0.0, and *metric*= *"cosine"*. Those are based on the commonly used metrics for NLP, *i.e.,* using *cosine* distance, and giving relevance to the local communities (*i.e.,* defined by *n_neighbors* parameter).
2. **Document clustering**: We used the *HDBSCAN* [5] algorithm with *min_cluster_size*=12, *metric*= *"euclidean"*, and *cluster_selection_method*= *"eom"* (Excess of Mass). It is a density-based clustering algorithm, identifying low-density instances as outliers based on the provided parameters. In our case, an outlier is a plugin that does not belong to any of the identified *dense* communities, having a minimum population size of 12.

[1] https://github.com/obsidianmd/obsidian-releases/blob/master/community-plugins.json
[2] https://huggingface.co/unsloth/gpt-oss-120b-GGUF
[3] https://maartengr.github.io/BERTopic/index.html
[4] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B-GGUF
[5] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

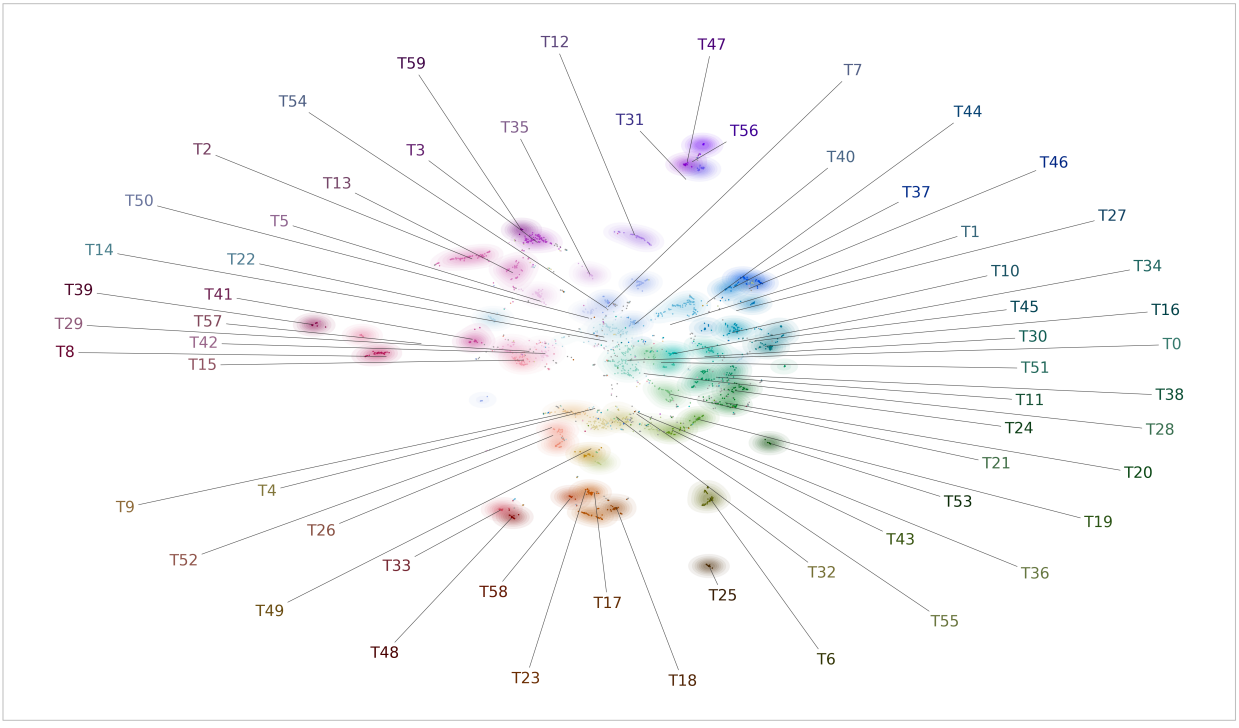


Figure 1: Two-dimensional UMAP projection of the saved plugin embeddings. Each point represents one plugin; color indicates its assigned leaf topic, and the leader-line identifiers T0–T59 map to the complete topic labels in Table 1. Gray points denote documents initially classified as outliers.

3. **Topic identification**: Each topic is composed using the default *BERTopic* token-based representation model, which uses weighted-keyword relevance among the documents to represent a single topic. We used a *Bag-of-Words* model, *CountVectorizer*[6], with English stop-word removal, bigram support (*i.e., ngram_range*=(1,2)), and a lemmatization preprocessor based on *WordNet*. The token weighting is then computed using the default *c-TF-IDF* algorithm[7].
4. **Topic labeling**: The *BERTopic* pipeline identified a set of distinct topics labeled by each topic's top terms. Each plugin result is assigned to a specific topic ID based on document relevance, except for the uncategorized plugins that are flagged as outliers.

After this, a postprocessing step has been applied to reduce the incidence of outlier plugins identified by *HDBSCAN*. To achieve this, we applied one of the outlier reduction techniques provided by *BERTopic*[8], specifically the topic distribution method. In detail, it assigns each outlier document to its most frequent topic, *i.e.,* the centroid closest to the topic, based on their distance in the embedding space.

We applied a semantic similarity threshold of 0.85 to keep the process conservative in terms of topic relevance. This threshold is in line with existing work that involves semantic similarity filtering [10, 16].

The final phase involves an automated LLM-based label generation to generate a descriptive and meaningful label for each identified topic. The prompts used are reported in the replication package [28]. We used the open-weight model *gpt-oss-120b* (F16 GGUF, *num_ctx*=8192, served through *Ollama*), with a parameter of *temperature*=0.5.

The labeling procedure works as follows. For each topic, five sample plugins were selected using the *BERTopic* built-in Maximal Marginal Relevance (MMR) algorithm to select the top-k representative documents for each topic. A diversity parameter of 0.3 was used, as per common practice. The representative plugin information is fed to the LLM, *i.e.,* plugin title, short description, and key phrases along with the assigned topic's top terms. The model was prompted

[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[7] https://maartengr.github.io/BERTopic/getting_started/ctfidf/ctfidf.html
[8] https://maartengr.github.io/BERTopic/getting_started/outlier_reduction/outlier_reduction.html

to generate a concise title label (at most five words). The previously assigned labels were also included in the prompt to encourage diversity and avoid redundancy. The topics were processed in a randomized sequence.

In addition, a second label has been generated, *i.e.,* a parent label, based on a higher-level grouping of similar identified topics obtained using the hierarchical topics function of *BERTopic*[9]. The function computes cosine distances between the c-TF-IDF topic representations and applies Ward hierarchical clustering with optimal leaf ordering. We visualized the resulting hierarchy as a dendrogram and used a distance cutoff of 1.0 to identify the parent groups.

Finally, for each parent topic, we generated a label using the same procedure as before, considering each parent topic as a distinct topic cluster and selecting as representative plugins a sample of five plugins among the child topics grouped by the parent. The previously generated topic label corresponding to each plugin is provided as additional input to the LLM.

3.3. Manual Validation

We conducted a stratified manual validation of the final taxonomy to assess three outputs on which the subsequent analyses depend: the LLM-generated keyphrases, the assigned leaf topic, and the assigned parent family. Before sampling, we excluded plugins assigned to the outlier topic (-1), records without valid leaf or parent labels, and duplicate plugin IDs or repositories. This produced an eligible pool of 2,381 plugins across 60 leaf topics. We ranked topics by eligible plugin count and selected the 20 largest, resolving equal counts by ascending topic ID. Using a fixed random seed of 42, we sampled 10 plugins without replacement from each selected topic and shuffled the resulting 200 records. The design therefore gives equal validation weight to each sampled major topic rather than allowing the largest topics to dominate the assessment.

Two authors independently examined each plugin's name, description, README, generated keyphrases, leaf-topic label, and parent-family label. For every plugin, each annotator answered the following binary questions using only *Yes* or *No*: (i) do the generated keyphrases represent the plugin? (ii) does the plugin fit its assigned leaf topic? and (iii) does the plugin fit its assigned parent family? We measured initial inter-annotator reliability using raw agreement and unweighted Cohen's κ. A third author then independently reviewed only the question-level disagreements and supplied the final decision for those items; agreements retained the shared answer of the two primary annotators.

For the adjudicated decisions, we report the number and proportion of accepted assignments together with two-sided Wilson 95% confidence intervals. The complete sample, the three annotation files, the disagreement-extraction script, the adjudicated dataset, and the analysis notebook are included in the replication package [28].

3.4. Ecosystem Structure Analysis

To characterize how the ecosystem is structurally organized, we generated three complementary topic-level views from the saved BERTopic artifacts, without refitting the topic model or recomputing plugin embeddings. The inter-topic distance map applies BERTopic's two-dimensional UMAP projection ($n_neighbors = 2$, cosine metric, random seed 42) to the 60 saved topic-embedding vectors; every circle represents a leaf topic and its area encodes the number of assigned plugins. To prevent label collisions, only the ten largest topics are named in the map, while every topic remains plotted. The similarity matrix reports cosine similarity among the 20 largest topic-embedding vectors, selected by descending plugin count and ascending topic ID to resolve ties; rows and columns are reordered into six similarity-based groups. The full dendrogram instead retains all 60 leaf topics and is computed from pairwise cosine distances between their c-TF-IDF representations using Ward linkage and optimal leaf ordering; branch colors and parent-label boxes show the groups obtained at the 1.0 cutoff.

Throughout the paper, *intra-family cohesion* refers only to semantic proximity in these model representations: leaf topics with more similar vectors have higher pairwise cosine similarity and merge at shorter dendrogram distances. It is descriptive shorthand for the geometry used to construct the hierarchy, not a separate cohesion coefficient, evidence of social coordination among developers, or independent validation of the six families. We therefore use the more explicit terms *within-family semantic proximity* and *between-family separation* in the summary claims.

The document map in Figure 1 was generated separately by applying UMAP with 10 neighbors, two components, a minimum distance of 0.0, cosine metric, and random seed 42 to the saved normalized plugin embeddings. Point color represents leaf-topic assignment; compact T0–T59 identifiers and leader lines avoid overlapping the full topic names, which are reported in Table 1. The word cloud in Figure 3 uses the concatenated documents assigned to non-outlier topics, with a deterministic layout (seed 42). Static publication figures were exported at 300 DPI. The complete

[9] https://maartengr.github.io/BERTopic/api/plotting/hierarchy.html

generation procedure and editable artifacts are provided in `5_generate_plots.ipynb` in the replication package [28].

3.5. Source Repositories Mining

To address RQ3, we mined the source software repositories for each plugin in our dataset. Since all repositories were hosted on GitHub, we used *PyGitHub*[10] to access the GitHub APIs and *PyDriller*[11] [31]to mine the commit histories. For each repository, we extracted: (i) the number of open, closed, and total pull requests; (ii) the number of open, closed, and total issues; (iii) the total number of commits (with first and last commit dates, using February 10, 2026 as a cutoff date); (iv) the number of stars; and (v) the number of GitHub Actions workflow definitions and recorded workflow runs by status (success, failure, cancelled, skipped). We use workflow presence as a proxy for repository automation and possible CI/CD use; because we did not classify the workflow YAML files by purpose, this measure does not establish that every workflow performs continuous integration or deployment. The indicators capture distinct observations: PR counts record proposed code changes; issue counts record reported bugs, requests, and other discussion items; commit counts and dates describe cumulative code-history activity and recency; stars approximate repository attention rather than maintenance; and GitHub Actions metrics describe configured and recorded automation. We use them as complementary descriptive views, not as interchangeable measures or as a composite ecosystem-health score. In particular, the open/closed fields do not encode the reason for closure or whether a PR was merged and an issue successfully resolved. Then, for each topic, we aggregated the obtained data by computing descriptive statistics (total, mean, open, and closed counts); Table 1 presents the resulting summary for pull requests and issues.

3.6. Development Practices Analysis

To characterize the engineering practices adopted in the ecosystem, we performed four complementary analyses using the repository-level metrics collected in the previous stage. First, we classified repositories as *active* (last commit within one year of our cutoff date) or *inactive*, and computed the distribution of repository creation dates by year to capture the ecosystem's growth trajectory. Second, we measured whether each repository contained at least one GitHub Actions workflow definition and aggregated the recorded run outcomes. We analyzed this binary workflow-presence variable directly in the correlation matrix, rather than inferring adoption from workflow or run counts. Neither the repository metadata nor this operationalization provides information about contributors' education, occupation, or professional-software-engineering experience. Third, we computed a Spearman rank-correlation matrix among eight repository-level metrics—stars, commits, pull requests, issues, binary GitHub Actions workflow presence, workflow count, recorded Actions-run count, and repository age—to reveal which dimensions of repository activity co-vary. Finally, we compared metric distributions across the parent topic groups identified in RQ1 to determine whether development practices differ systematically by functional domain.

4. Results

This section presents the findings from our analysis of the *Obsidian* plugin ecosystem, structured around four research questions. *RQ1* examines what kinds of plugins exist and how they are functionally organized. *RQ2* investigates how the ecosystem is structurally organized in terms of inter-topic relationships. *RQ3* assesses the extent to which plugins are actively maintained and evolved. *RQ4* characterizes the development practices adopted across the ecosystem.

4.1. RQ1: Plugin Topics in the Obsidian Ecosystem

Our analysis of the complete Obsidian plugin ecosystem shows that it is both diverse and semantically coherent, producing a total of 60 different leaf topics organized into approximately 15 parent groups at a hierarchy cutoff distance of 1.0, which in turn form six broad thematic families.

Figure 1 displays the two-dimensional projection of plugin embeddings, where clusters reveal distinct functional areas. The most densely populated regions correspond to productivity-centric workflows—*Daily Notes Automation*, *Comprehensive Task Management*, *Integrated Timeline Calendar Suite*, *Grammar and Spellchecking Tools*, and *Template-Based Note Generation*—highlighting Obsidian's role as a workspace for knowledge-centric workflows rather than software development.

[10] https://github.com/PyGitHub/PyGitHub
[11] https://github.com/ishepard/pydriller

T	N	Topic Label	Parent Label	PRs Tot.	PRs Mean	PRs Open	PRs Closed	Iss. Tot.	Iss. Mean	Iss. Open	Iss. Closed
59	14	Todoist Integration Suite	Comprehensive Task Mgmt.	346	24.71	14	332	514	36.71	168	346
3	86	Comprehensive Task Management	Comprehensive Task Mgmt.	**3806**	**44.26**	**167**	**3639**	**6032**	70.96	2131	**3901**
2	83	Habit and Time Tracker	Daily Note Prod. Suite	1800	21.69	66	1734	2408	29.01	809	1599
13	53	Integrated Timeline Calendar Suite	Daily Note Prod. Suite	944	17.81	102	842	2257	43.40	864	1393
5	75	Daily Notes Automation	Daily Note Prod. Suite	868	11.57	44	824	1226	16.35	473	753
50	31	Recent Activity Tracker	Daily Note Prod. Suite	266	8.58	18	248	878	28.32	263	615
52	19	Annotated Article Sync	Academic Annot. Workflow	536	28.21	18	518	426	22.42	170	256
26	33	Book Highlights Import	Academic Annot. Workflow	1089	33.00	44	1045	1262	38.24	393	869
49	24	Comprehensive PDF Toolkit	Academic Annot. Workflow	601	25.04	38	563	1536	64.00	681	855
25	29	Research Citation Management	Academic Annot. Workflow	206	7.10	39	167	864	32.00	423	441
33	25	Tabletop RPG Companion Suite	Learning Reference Suite	756	30.24	34	722	967	38.68	304	663
8	49	AI-Enhanced Flashcard Sync	Learning Reference Suite	883	18.02	82	801	1898	38.73	969	929
39	21	Bible Reference Integration	Learning Reference Suite	443	21.10	21	422	471	22.43	95	376
31	30	Audio Note Processing	Learning Reference Suite	403	13.43	21	382	791	26.37	269	522
47	21	YouTube Video Integration	Learning Reference Suite	226	10.76	35	191	890	42.38	267	623
57	20	Multilingual Vocabulary Builder	Writing Productivity Suite	148	7.40	21	127	396	19.80	145	251
41	21	Grammar and Spellchecking Tools	Writing Productivity Suite	114	5.43	6	108	262	12.48	70	192
22	34	Productive Writing Workspace	Writing Productivity Suite	121	3.56	23	98	707	20.79	287	420
29	52	AI Writing Companion	AI-Powered Vault Assistant	737	14.45	32	705	1172	22.98	330	842
15	64	AI Note Chat Integration	AI-Powered Vault Assistant	2583	40.36	106	2477	3209	50.14	823	2386
42	47	Semantic AI Vault Assistant	AI-Powered Vault Assistant	939	19.98	39	900	1413	30.06	402	1011
23	32	Canvas Interaction Enhancements	Canvas Productivity Toolkit	**39**	**1.22**	7	**32**	209	6.74	81	128
58	14	Interactive Markdown Mindmaps	Canvas Productivity Toolkit	228	16.29	19	209	1555	**111.07**	924	631
54	18	Advanced Obsidian Data Queries	Table Management Suite	335	18.61	17	318	383	21.28	142	241
35	26	Advanced Table Management	Table Management Suite	142	5.46	14	128	993	38.19	373	620
6	56	Comprehensive LaTeX Math Suite	Interactive Visual Embedding	660	11.79	54	606	1474	26.80	502	972
17	41	Interactive Graph Visualization	Interactive Visual Embedding	546	13.32	32	514	997	24.32	415	582
18	41	Dynamic Diagram Rendering	Interactive Visual Embedding	775	18.90	77	698	2756	67.22	1247	1509
48	19	Board Game Diagram Rendering	Interactive Visual Embedding	202	10.63	6	196	**121**	**6.37**	**62**	**59**
56	17	Chord Notation Toolkit	Interactive Visual Embedding	49	2.88	**5**	44	322	18.94	124	198
19	37	Enhanced Code Block Features	Code Block Enh. Toolkit	698	18.86	23	675	430	11.62	184	246
43	25	Embedded Programming Sandbox	Code Block Enh. Toolkit	389	15.56	29	360	570	22.80	259	311
53	17	Plugin Mgmt. & Version Control	Code Block Enh. Toolkit	196	11.53	11	185	405	23.82	123	282
55	18	Image and PDF OCR	Automated Image Mgmt.	460	25.56	26	434	655	36.39	175	480
44	28	Media Gallery Management	Automated Image Mgmt.	124	4.43	20	104	662	23.64	305	357
46	25	Advanced Image Conversion Suite	Automated Image Mgmt.	206	8.24	26	180	837	33.48	319	518
37	27	Clipboard Image Auto Upload	Automated Image Mgmt.	577	21.37	22	555	476	17.63	149	327
21	33	Smart Clipboard Copy Suite	Adv. Note Linking Suite	167	5.06	18	149	330	10.00	124	206
28	36	Markdown Layout and Annotations	Adv. Note Linking Suite	280	7.78	27	253	948	27.09	415	533
0	107	Advanced Link Management	Adv. Note Linking Suite	636	5.94	62	574	2012	18.80	778	1234
32	27	Rich External File Integration	Adv. Note Linking Suite	187	6.93	36	151	1047	38.78	521	526
9	52	Obsidian Blog Publishing	Adv. Note Linking Suite	581	11.17	19	562	1113	21.82	433	680
4	97	Cross-Platform Note Publishing	Adv. Note Linking Suite	1322	13.63	106	1216	3300	34.02	1064	2236
1	110	Vault Management & Security	Adv. Note Linking Suite	1910	17.36	127	1783	5884	53.49	**2156**	3728
12	45	GitHub and Jira Issue Integration	Adv. Note Linking Suite	1728	38.40	76	1652	1519	33.76	417	1102
10	56	File Explorer Enhancements	File Mgmt. Enhancements	210	3.75	49	161	1158	20.68	473	685
27	27	Intelligent Filename Sync.	File Mgmt. Enhancements	225	8.33	21	204	559	20.70	183	376
40	24	Automated Attachment Folder Mgmt.	File Mgmt. Enhancements	239	9.96	27	212	382	15.92	212	170
45	26	Dynamic UI Theme Customization	Contextual Shortcut Autom.	406	15.62	26	380	578	22.23	232	346
36	42	Productivity Shortcuts Automation	Contextual Shortcut Autom.	204	4.86	16	188	532	12.67	178	354
24	33	Custom Editor Shortcut Suite	Contextual Shortcut Autom.	296	8.97	23	273	1214	37.94	520	694
38	23	Cursor Scroll Enhancements	Contextual Shortcut Autom.	52	2.26	15	37	147	6.39	71	76
14	108	Template-Based Note Generation	Contextual Shortcut Autom.	1126	10.43	93	1033	3360	31.11	927	2433
30	40	Streamlined Note Management	Contextual Shortcut Autom.	134	3.35	21	113	580	14.50	268	312
16	42	Browser-Style Tab Controls	Contextual Shortcut Autom.	127	3.02	13	114	814	19.38	287	527
7	54	Intelligent Tagging Suite	Enhanced Text & Tag Edit.	380	7.04	26	354	1114	20.63	451	663
20	35	Enhanced Text Formatting	Enhanced Text & Tag Edit.	1516	44.59	42	1474	1840	54.12	514	1326
11	46	Regex-Based Text Highlighting	Enhanced Text & Tag Edit.	163	3.54	16	147	799	17.76	331	468
51	17	Callout and List Customization	Enhanced Text & Tag Edit.	107	6.29	9	98	514	30.24	150	364
34	29	Heading Visualization & Navigation	Enhanced Text & Tag Edit.	290	10.00	20	270	1069	36.86	257	812

Table 1: Summary of topics with pull request and issue metrics. Blue indicates the maximum value in a column; red indicates the minimum value. Parent label colours match the cluster colours in Figure 2.

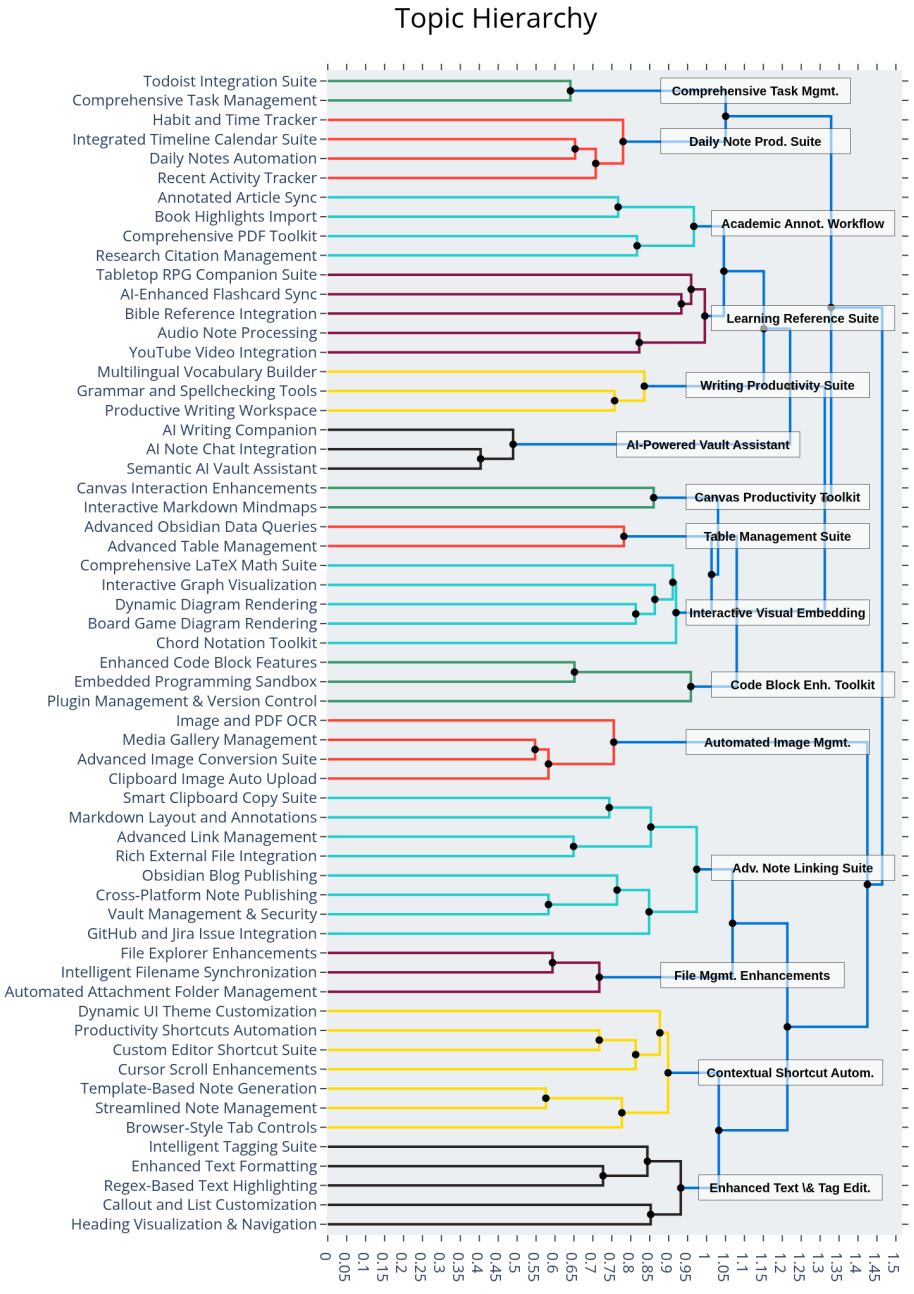


Figure 2: Full 60-topic hierarchy based on Ward clustering of cosine distances between c-TF-IDF topic representations. The horizontal axis reports merge distance: topics joined farther to the left are more similar. Leaf labels are aligned with their branches; branch colors and labeled boxes identify the parent groups obtained at the 1.0 cutoff.

More specialized or creative functionalities appear in clearly separated peripheral clusters, such as *Dynamic Diagram Rendering*, *Board Game Diagram Rendering*, *Chord Notation Toolkit*, *Tabletop RPG Companion Suite*, and *Research Citation Management*. Although smaller in size, these groups appear as distinct neighborhoods in the two-dimensional projection, illustrating extensions for diverse and even niche use cases—from music notation to tabletop gaming—within a knowledge-centric community. Notably, a distinct cluster of AI-related plugins has emerged on



Figure 3: Word cloud constructed from the concatenated documents assigned to the 60 non-outlier topics. Font area encodes corpus term frequency; the layout is fixed with random seed 42.

the left periphery of the embedding space, including *AI Writing Companion*, *AI Note Chat Integration*, *Semantic AI Vault Assistant*, and *AI-Enhanced Flashcard Sync*, reflecting the growing integration of large language models into knowledge-management workflows.

The hierarchical perspective in Figure 2 further illustrates how these 60 topics relate at broader levels of abstraction. Colored brackets identify parent groups that merge at different distances; topics that merge at shorter distances are more semantically similar. Six broad thematic families emerge from this hierarchy: (i) **Task & productivity management**: *Todoist Integration Suite*, *Comprehensive Task Management*, *Habit and Time Tracker*, *Integrated Timeline Calendar Suite*, *Daily Notes Automation*, and *Recent Activity Tracker* merge into a single family of workflow-support tools. (ii) **Content import & reference**: *Book Highlights Import*, *Comprehensive PDF Toolkit*, *Research Citation Management*, *Annotated Article Sync*, *AI-Enhanced Flashcard Sync*, *Bible Reference Integration*, *Audio Note Processing*, *YouTube Video Integration*, and *Multilingual Vocabulary Builder* form a domain centered on knowledge acquisition from external sources. (iii) **Writing & AI assistance**: *Grammar and Spellchecking Tools*, *Productive Writing Workspace*, *AI Writing Companion*, and *AI Note Chat Integration* constitute a cluster, with *Semantic AI Vault Assistant* nearby. (iv) **Visualization & technical tools**: *Canvas Interaction Enhancements*, *Interactive Markdown Mindmaps*, *Comprehensive LaTeX Math Suite*, *Interactive Graph Visualization*, *Dynamic Diagram Rendering*, *Board Game Diagram Rendering*, *Chord Notation Toolkit*, *Enhanced Code Block Features*, and *Embedded Programming Sandbox* form a distinct branch spanning data visualization, diagramming, and code execution. (v) **File management & publishing**: *Obsidian Blog Publishing*, *Cross-Platform Note Publishing*, *Vault Management & Security*, *GitHub and Jira Issue Integration*, *File Explorer Enhancements*, *Intelligent Filename Synchronization*, and *Automated Attachment Folder Management* compose a group centered on vault organization, security, and content publishing. (vi) **Editor customization**: *Dynamic UI Theme Customization*, *Productivity Shortcuts Automation*, *Custom Editor Shortcut Suite*, *Cursor Scroll Enhancements*, *Browser-Style Tab Controls*, *Heading Visualization & Navigation*, *Enhanced Text Formatting*, *Regex-Based Text Highlighting*, and *Callout and List Customization* cluster together, covering interface appearance, keyboard workflows, and text formatting. Additional topics such as *Advanced Link Management*, *Rich External File Integration*, *Markdown Layout and Annotations*, *Plugin Management & Version Control*, and image-related plugins (*Media Gallery Management*, *Advanced Image Conversion Suite*, *Clipboard Image Auto Upload*) bridge multiple families, acting as cross-cutting connectors within the ecosystem.

The *Learning Reference Suite* parent group raised a less immediate relationship than groups centered on a single artifact or activity. Its commonality is at the workflow level: flashcard and vocabulary plugins support learning; Bible and tabletop-role-playing plugins provide structured reference material; and audio- and video-processing plugins bring external learning or reference sources into the vault. Thus, the label denotes support for consulting, capturing, or rehearsing domain knowledge rather than similarity of the source media.

Figure 3 displays a word cloud of the aggregated topic vocabulary. The dominant terms—*note*, *file*, *markdown*, *link*, *vault*—confirm that the ecosystem revolves around knowledge artifacts and personal information management, rather than programming constructs. Secondary terms such as *integration*, *custom*, *template*, *image*, and *task* reflect the breadth of workflow automation and content enrichment that plugins provide.



Figure 4: Word cloud of the vocabulary for documents initially classified as outliers before the distribution-based reassignment strategy was applied.

Dimension	Agreement	Kappa	Accepted	Rate (95% CI)
Keyphrases	100.0%	–	200/200	100.0% (98.1%–100.0%)
Leaf topic	96.0%	0.647	191/200	95.5% (91.7%–97.6%)
Parent family	94.0%	0.615	186/200	93.0% (88.6%–95.8%)

Table 2: Manual-validation agreement and adjudicated acceptance rates. Acceptance rates include two-sided Wilson 95% confidence intervals.

Comparing the main word cloud with that of the outlier documents (Figure 4)—i.e., plugins that resisted initial cluster assignment—reveals a shift in vocabulary emphasis. Outlier plugins are characterized by terms such as *command*, *custom*, *auto*, *search*, *editing*, and *view*, suggesting they tend to be generic utility or editor-enhancement plugins that do not align neatly with a single thematic family. Their vocabulary overlaps with multiple families, which explains why the clustering algorithm initially left them unassigned and why the outlier-reduction strategy was necessary to incorporate them.

Manual-validation results. Table 2 reports the initial agreement between the two primary annotators and the final acceptance rates after adjudication. The annotators agreed on all 200 keyphrase decisions, on 192 of 200 leaf-topic decisions (96.0%), and on 188 of 200 parent-family decisions (94.0%). Cohen's κ was 0.647 for leaf topics and 0.615 for parent families; it is undefined for keyphrases because both annotators used the same category (*Yes*) for every plugin. After the third annotator resolved the disagreements, 200 keyphrase sets (100.0%), 191 leaf assignments (95.5%), and 186 parent assignments (93.0%) were accepted. These results provide strong human-assessed content validity for the sampled major topics while also showing that parent-family assignment is the most difficult of the three decisions.

Answer to RQ1

We identified 60 leaf topics organized into approximately 15 parent groups and six broad thematic families: task & productivity management, content import & reference, writing & AI assistance, visualization & technical tools, file management & publishing, and editor customization. The ecosystem vocabulary centers on knowledge artifacts (notes, files, markdown, links) rather than programming constructs. A distinct AI-related cluster reflects the growing adoption of LLMs in knowledge-management workflows, while outlier plugins tend to be generic utilities that span multiple families. A stratified manual assessment of 200 plugins from the 20 largest topics accepted 100.0% of keyphrase sets, 95.5% of leaf-topic assignments, and 93.0% of parent-family assignments after adjudication.

4.2. RQ2: Ecosystem Structure and Inter-Topic Relationships

Beyond identifying individual topics, we analyze three structural views of the ecosystem to understand how topics relate to one another.

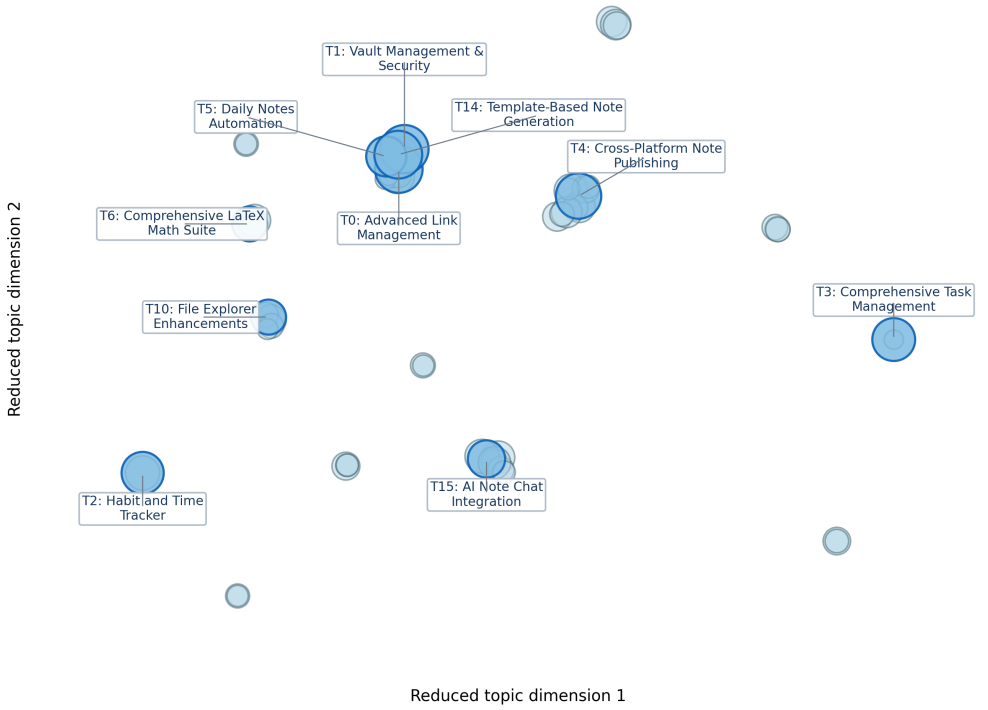


Figure 5: Static inter-topic distance map based on a two-dimensional UMAP projection of BERTopic topic embedding vectors. Each circle is a leaf topic and its area is proportional to plugin count. For legibility, only the ten largest topics are labelled; their T identifiers map to Table 1, and the unlabelled circles retain the other 50 topics. Greater separation indicates lower proximity in the two-dimensional projection.

Figure 5 presents the intertopic distance map. The labelled circles make the size imbalance explicit: *Vault Management & Security* (110 plugins), *Template-Based Note Generation* (108), and *Advanced Link Management* (107) are the largest topics, whereas most circles are substantially smaller. We describe this as a long-tailed distribution within Obsidian; without equivalently sampled data and the same topic-size definition, we do not use it as quantitative evidence of similarity to another ecosystem. Nearby and isolated circles suggest relationships for follow-up, but we do not interpret UMAP quadrants as categorical boundaries; the similarity matrix and full hierarchy provide the direct pairwise and hierarchical evidence.

Figure 6 presents pairwise cosine similarity for the 20 largest topic representations. Off-diagonal values range from 0.737 to 0.953. The matrix recovers close within-domain relationships, including *Cross-Platform Note Publishing* with *Advanced Link Management* (0.930) and *Comprehensive Task Management* with *Habit and Time Tracker* (0.900), while lower values involving *Productivity Shortcuts Automation*—for example, 0.737 with *Habit and Time Tracker*—show separation within the major-topic subset. Because the displayed set is selected by size, the dendrogram in Figure 2 remains the granular reference for relationships involving all 60 topics.

The hierarchical dendrogram (Figure 2) complements this view: tightly related topics merge at distances below 0.85 (e.g., *Habit and Time Tracker* with *Integrated Timeline Calendar Suite*), while the last merges—connecting the six thematic families—occur only above distance 1.0. Under our operational definition, the shorter topic-level linkages represent within-family semantic proximity, whereas the family-level merges above 1.0 represent between-family separation. Because the families are derived from this same hierarchy, this geometry describes the model's organization and is not independent validation of the grouping.

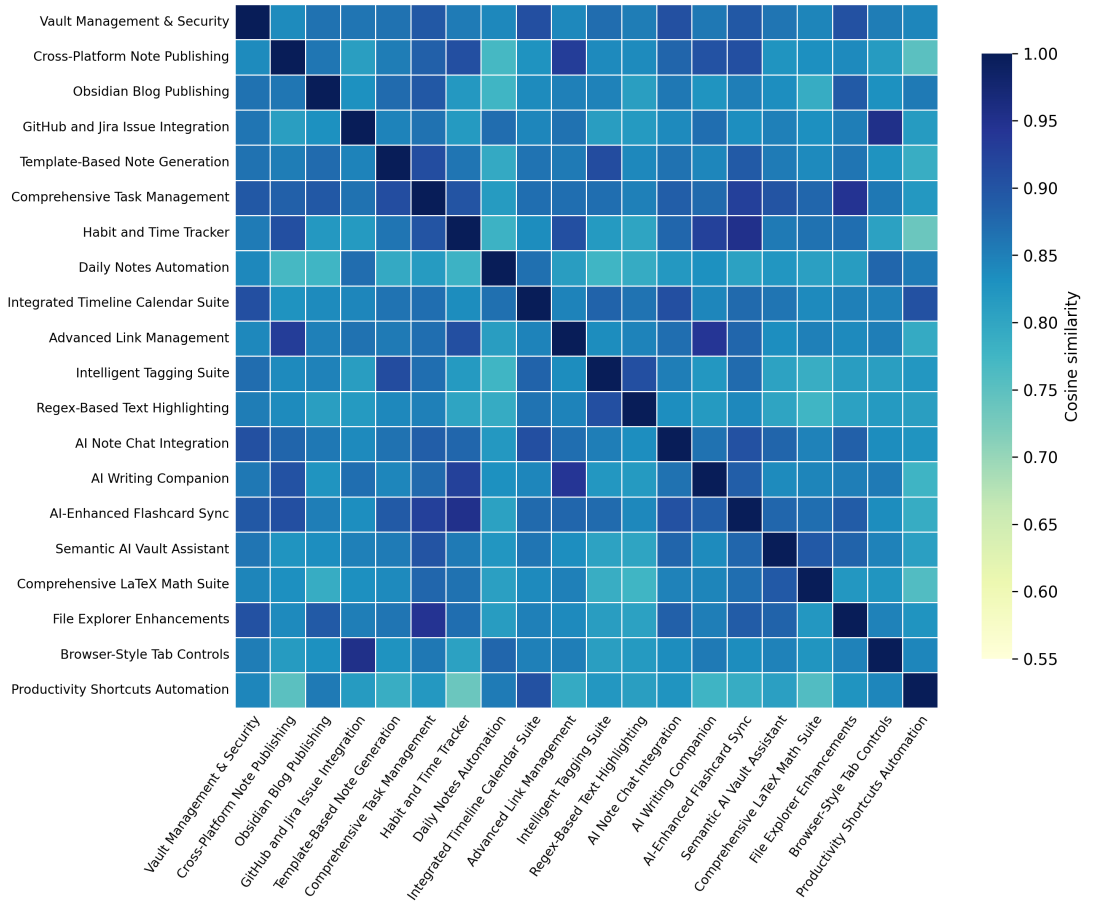


Figure 6: Pairwise cosine-similarity matrix for the 20 largest leaf-topic embedding vectors, reordered into six similarity-based groups. The same full labels are aligned on the rows and columns; darker cells indicate higher similarity. The complete 60-topic hierarchy is retained in Figure 2.

Answer to RQ2

The Obsidian plugin ecosystem exhibits a modular, long-tailed structure. A few large topics dominate in size, while the majority are small, specialized niches. Among the 20 largest topics, cosine similarities span 0.737–0.953: close within-domain relationships coexist with cross-cutting semantic overlap. The dendrogram shows the representation-space separation used to obtain the six thematic families, merging only at distances above 1.0.

4.3. RQ3: Maintenance and Evolution of the Plugins

Table 1 shows pull-request (PR) and issue activity aggregated per topic. The results reveal substantial variation in maintenance intensity but overall consistent evidence of recorded repository activity across the ecosystem.

High-activity topics such as *Comprehensive Task Management* (3,806 PRs, 6,032 issues) and *AI Note Chat Integration* (2,583 PRs, 3,209 issues) stand out with the highest total volumes of both PRs and issues. The issue dimension highlights additional patterns: *Vault Management & Security* accumulates the most open issues (2,156), while *Interactive Markdown Mindmaps* achieves the highest mean issue count per plugin (111.07), showing high recorded issue volume relative to its topic size. In contrast, more specialized or creative categories such as *Chord Notation Toolkit* and *Board Game Diagram Rendering* show lower PR and issue volumes, consistent with their smaller topic sizes and narrower functional scope.

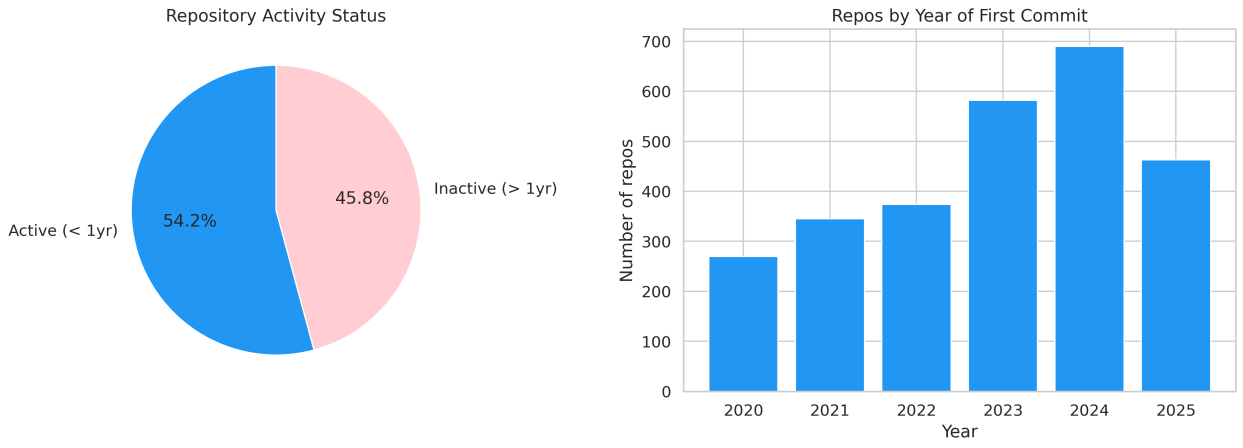


Figure 7: Repository activity status and ecosystem growth. Left: proportion of active (last commit within one year) vs. inactive repositories. Right: number of plugin repositories by year of first commit.

Across nearly all topics, closed PR and issue counts exceed open counts, providing evidence of substantial historical processing activity. These aggregate states do not reveal whether a closed PR was merged or rejected, whether a closed issue was fixed, duplicated, or declined, how quickly items were handled, or the demographics of the contributors. We therefore interpret them as repository-activity indicators rather than direct measures of responsiveness or comparisons with other ecosystems. Notably, the structural modularity revealed in RQ2 is also reflected in the maintenance data: high-activity topics belong predominantly to the productivity and content-import families—the largest in terms of plugin count—while niche families such as visualization and music notation show sparser but still present maintenance traces.

Answer to RQ3

The results show substantial variation in maintenance intensity and widespread recorded activity across pull requests and issues.
Closed counts exceed open counts in nearly all topics; this documents historical processing volume but does not by itself establish response quality or contributor demographics. High-activity topics (e.g., task management, AI chat integration) contrast with niche categories (e.g., chord notation, board-game rendering). The activity distribution mirrors the structural modularity of the ecosystem.

4.4. RQ4: Development Practices and Repository Activity

Beyond PR and issue activity, we describe repository growth, GitHub Actions workflow presence and recorded runs, metric inter-relationships, and cross-topic variation. We avoid treating these repository proxies as measurements of contributor demographics or overall engineering maturity.

Ecosystem growth and activity. Figure 7 presents two complementary views of ecosystem vitality. The pie chart shows that 54.2% of plugin repositories have received at least one commit within the last year (i.e., after February 2025), while 45.8% have been inactive for more than one year.

The bar chart traces the number of repositories by the year of their first commit: annual counts rise from 270 in 2020 to a peak of 690 in 2024 and then fall to 463 in the complete 2025 calendar year. Because these are repository first-commit counts rather than contributor records, they describe registry growth but not how many distinct developers joined. The activity split is also descriptive: "inactive" means only that no commit was recorded within the one-year window and cannot distinguish stable software from abandonment. We consequently make no cross-ecosystem or demographic inference from either panel.

GitHub Actions workflow presence and recorded outcomes.

Figure 8 shows that 1,660 of 2,724 unique plugin repositories (60.9%) contain at least one GitHub Actions workflow definition. We treat this as an automation/CI-CD proxy, not verified CI/CD adoption, because workflow purpose was

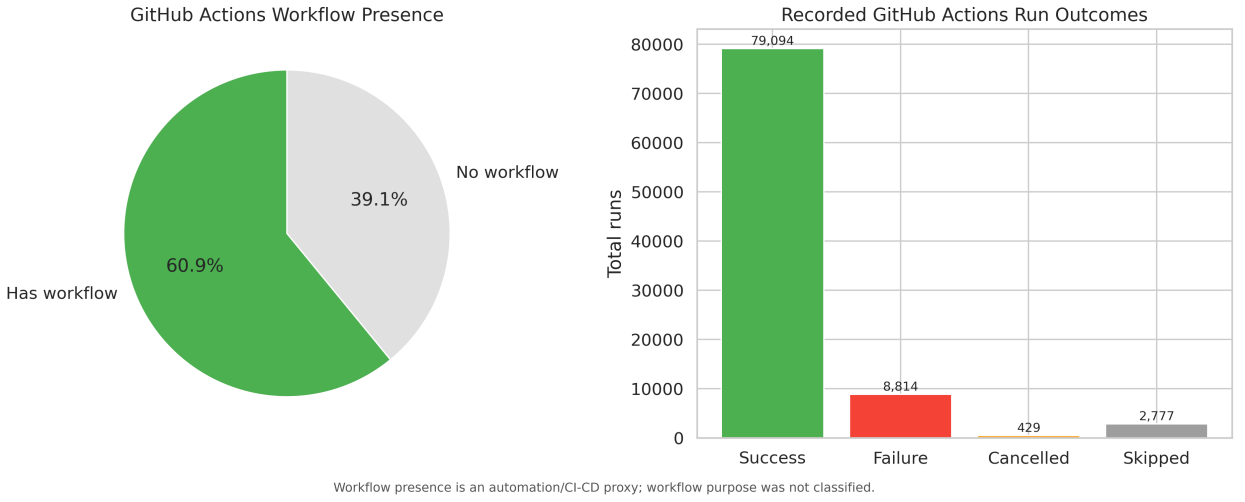


Figure 8: GitHub Actions workflow presence and recorded run outcomes. Left: repositories with at least one workflow definition; this is an automation/CI-CD proxy because workflow purpose was not classified. Right: the four displayed recorded run-outcome categories across workflow-bearing repositories.

not classified. Among the four displayed recorded outcome categories, 79,094 of 91,114 runs (86.8%) completed successfully, 8,814 (9.7%) failed, 2,777 (3.0%) were skipped, and 429 (0.5%) were cancelled.

Among the 1,068 workflow-bearing repositories for which at least one run was recorded, the median per-repository proportion of successful recorded runs is 0.95. A successful Actions invocation does not by itself establish build quality, deployment, or software reliability. Workflow-bearing repositories define a median of one workflow.

GitHub Actions workflow presence varies across topic categories. Figure 9 shows that *Academic Annotation Workflow* leads with 71% workflow presence ($n = 105$), followed by *Daily Note Productivity Suite* (69%, $n = 242$) and *Comprehensive Task Management* (66%, $n = 100$). At the lower end, *Canvas Productivity Toolkit* (52%, $n = 46$) and *File Management Enhancements* (52%, $n = 107$) show the lowest workflow-presence rates, though still above half of all repositories.

The *AI-Powered Vault Assistant* category has 66% workflow presence ($n = 163$); the repository data do not support attributing this observation to developer experience.

Metric inter-relationships. Figure 10 presents the Spearman rank-correlation matrix among eight repository-level metrics. The strongest correlation is between *Stars* and *Issues* ($\rho = 0.83$), showing that their repository-level ranks co-vary in this snapshot; the association does not establish that stars cause issue activity or identify the people creating either signal. *Stars* also correlate moderately with *PRs* ($\rho = 0.57$) and *Commits* ($\rho = 0.51$), suggesting that popular plugins attract not only user attention but also development contributions.

Workflow count and recorded Actions runs are strongly correlated ($\rho = 0.72$). More importantly for the adoption interpretation, the binary workflow-presence variable has only a weak positive association with stars ($\rho = 0.10$) and a negligible association with repository age ($\rho = -0.05$). It is also weakly associated with commits ($\rho = 0.26$), PRs ($\rho = 0.27$), and issues ($\rho = 0.15$). These estimates do not support claims that workflow presence is independent of popularity or that newer repositories are substantially more likely to adopt it.

Cross-topic variation in development metrics. Figure 11 shows the distribution of four key metrics—stars, commits, issues, and recorded Actions runs—across the top 15 parent topic categories by plugin count. *Table Management Suite* achieves the highest median star count, followed by *Academic Annotation Workflow* and *AI-Powered Vault Assistant*, suggesting that these topics serve highly valued niches within the community. In terms of development intensity, *Academic Annotation Workflow* and *Comprehensive Task Management* lead in median commits, while *Comprehensive Task Management* also leads in issue counts, reflecting complex functionality that demands both active development and extensive user feedback.

Recorded Actions-run counts are highest for *Academic Annotation Workflow* and *AI-Powered Vault Assistant*, alongside their higher workflow-presence rates. Notably, all topic categories show wide interquartile ranges, indicating

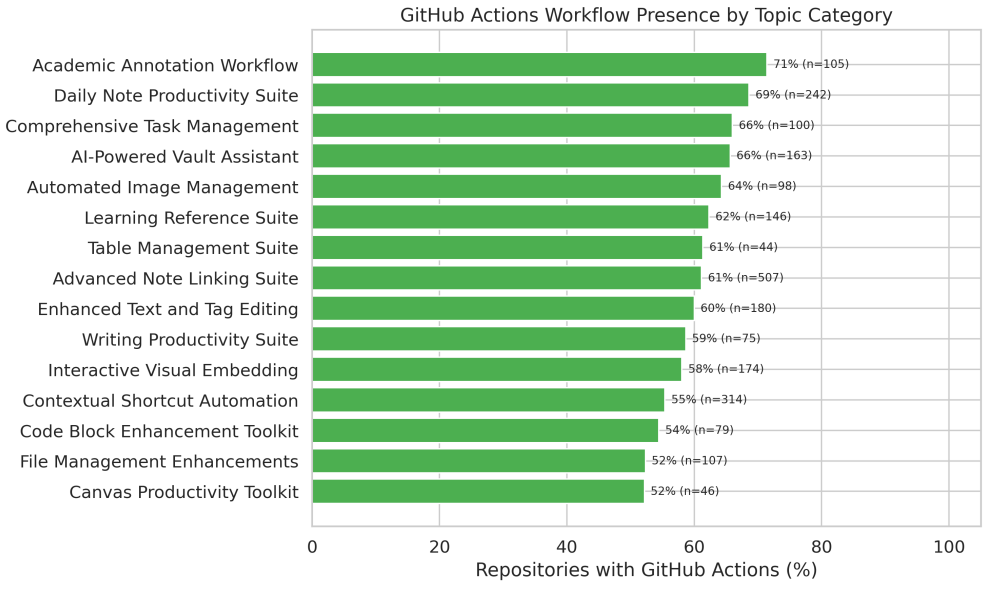


Figure 9: GitHub Actions workflow presence by parent topic category. Bars show the percentage of repositories with at least one GitHub Actions workflow; *n* indicates the total number of repositories in each category.

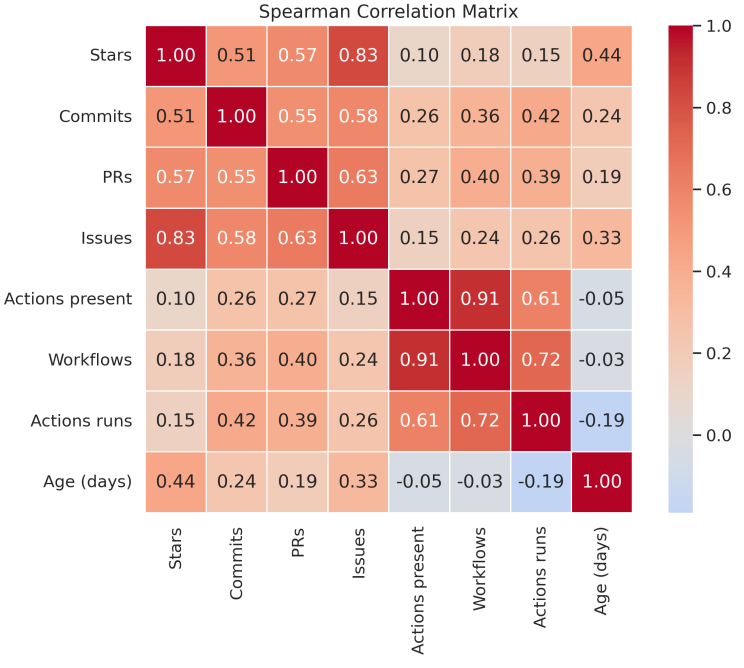


Figure 10: Spearman rank-correlation matrix among eight repository-level metrics. GitHub Actions presence is the binary workflow-presence proxy; workflow and run counts are separate activity measures. Red indicates positive and blue negative association.

substantial within-topic heterogeneity: even within a single functional domain, plugin repositories span a wide range of recorded development activity and community attention.

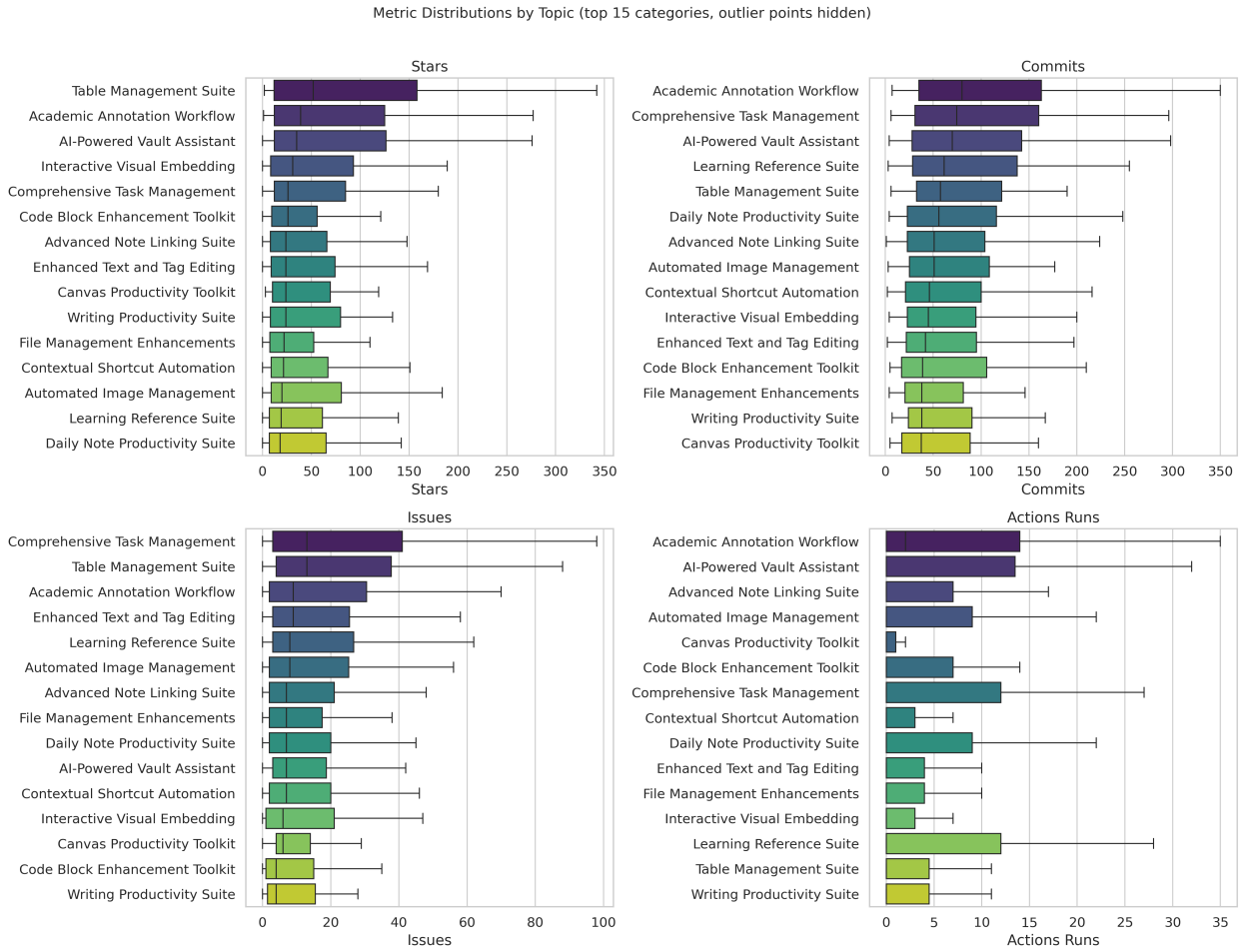


Figure 11: Distribution of development metrics (stars, commits, issues, recorded GitHub Actions runs) across the top 15 parent topic categories. Box plots show medians and interquartile ranges; outlier points are hidden for clarity.

Answer to RQ4

At the cutoff, 54.2% of repositories have a commit within the preceding year. Of 2,724 unique repositories, 1,660 (60.9%) contain a GitHub Actions workflow; among the 1,068 with recorded runs, the median observed success proportion is 0.95. Stars and issues are strongly correlated ($\rho = 0.83$). Workflow presence is weakly associated with stars ($\rho = 0.10$) and negligibly associated with repository age ($\rho = -0.05$), so the data do not establish popularity independence or a generational adoption trend. Repository metrics vary substantially within and across topic categories.

5. Discussion

We discuss the key findings and their implications for the software engineering community.

5.1. A Structured Ecosystem Beyond Developer Communities

Our results reveal that the Obsidian plugin ecosystem is far from a disorganized catalogue of ad-hoc extensions. The 60-topic taxonomy is structured into six thematic families (RQ1), with within-family semantic proximity and between-family separation in the model representations (RQ2), and recorded repository activity across all families (RQ3). This structural regularity is remarkable given that the ecosystem was not planned top-down by a platform

vendor, but emerged organically from a community whose primary concern is knowledge management rather than software development.

The similarity matrix (Figure 6) and hierarchical dendrogram (Figure 2) identify semantic functional boundaries in the topic representations; they do not measure social coordination among developers. Productivity topics (task management, daily notes, calendars) form a tight cluster that rarely overlaps with visualization topics (diagrams, graphs, math) or editor-customization topics (themes, shortcuts, tab controls). This modular structure is consistent with prior work showing that modular architectures enable distributed innovation and reduce coordination costs in large ecosystems [3, 17].

Within Obsidian, a handful of broad topics dominate in plugin count while a tail of niche extensions covers uses from tabletop RPGs to music notation. Prior research on developer-facing marketplaces motivates asking whether such structures recur elsewhere, but this study has not derived equivalent topic distributions from those ecosystems and therefore does not claim empirical equivalence.

5.2. The Emergence of AI-Powered Knowledge Plugins

Perhaps the most striking finding is the emergence of a distinct AI-related cluster on the periphery of the embedding space (Figure 1). Plugins such as *AI Writing Companion*, *AI Note Chat Integration*, *Semantic AI Vault Assistant*, and *AI-Enhanced Flashcard Sync* collectively form a new thematic area that did not exist in earlier snapshots of the ecosystem. This cluster signals that generative-AI capabilities are rapidly permeating knowledge-centric communities, blurring the boundary between users and developers even further. Unlike traditional developer-tool integrations (e.g., code completion for IDEs), these AI plugins augment *thinking tasks*: summarizing notes, generating flashcards from knowledge bases, and enabling conversational interaction with personal vaults. From an end-user development perspective, these plugins can be understood as mechanisms that enable non-programmers to construct and adapt computational artifacts [13, 14, 22].

This trend has significant implications for software engineering research. First, it challenges the assumption that LLM-based tooling is primarily relevant to programming workflows [9, 18]; in knowledge-centric ecosystems, LLMs become mediation tools between users and their own knowledge artifacts. Second, it raises new questions about trust, transparency, and evaluation in non-developer contexts [1, 7, 23, 30]: how do non-technical users evaluate the quality of AI-generated summaries or flashcards, and how should plugin developers communicate model limitations?

5.3. Vocabulary as an Ecosystem Fingerprint

The ecosystem vocabulary, dominated by terms such as *note*, *file*, *markdown*, and *link* (Figure 3), provides a concise fingerprint of the community's primary concern: knowledge artifacts and their relationships, rather than code constructs. This aligns with models of information interaction, where users iteratively organize, filter, and retrieve information to support cognitive tasks [19, 25]. The contrast with the outlier word cloud (Figure 4)—where terms such as *command*, *auto*, *search*, and *editing* dominate—suggests that generic utility plugins act as the "connective tissue" of the ecosystem, serving multiple families without belonging to any single one. A comparative vocabulary analysis with developer-centric ecosystems (e.g., extracting dominant terms from VS Code extension descriptions) could quantify how knowledge-centric and developer-centric communities differ in their functional priorities and would contribute to an operational definition of "hybrid ecosystems."

5.4. Maintenance in a Knowledge-Centric Platform

The maintenance metrics (RQ3) show recorded repository activity across all topic families. Closed PR and issue counts exceed open counts in nearly every topic, and even niche topics such as *Chord Notation Toolkit* and *Board Game Diagram Rendering* have recorded activity. These state counts cannot distinguish merged from rejected PRs, resolved from declined issues, or stable from abandoned repositories. Moreover, GitHub repository metadata do not identify contributors' occupations, training, or experience. "Knowledge-centric" therefore characterizes the platform and plugin functionality, not a measured developer demographic; surveys or contributor-profile studies would be required to test demographic or motivational explanations.

Larger topic families have higher aggregate PR volumes, but topic size mechanically contributes to those totals; Table 1 therefore also reports per-plugin means. Neither view identifies the quality or outcome of a contribution, community demand, developer investment, or future abandonment. Evaluating those mechanisms would require closure-reason and merge-status classification, response-time measures, release histories, and contributor or user evidence beyond the present repository snapshot.

5.5. Repository Automation in a Knowledge-Centric Ecosystem

The development-practices analysis (RQ4) provides an ecosystem-internal description of repository automation. Of 2,724 unique repositories, 1,660 (60.9%) contain at least one GitHub Actions workflow. This prevalence is not compared numerically with unmatched open-source populations: repository selection, observation periods, and definitions of CI/CD adoption differ, and the present data contain no contributor demographics. Furthermore, because workflow definitions were counted but not classified by purpose, workflow presence is only an automation/CI-CD proxy. Among the 1,068 workflow-bearing repositories with recorded runs, the median observed success proportion is 0.95; this describes recorded Actions outcomes, not verified build or release reliability.

The Spearman analysis (Figure 10) separates the binary presence of a workflow from workflow and run counts. Stars and issues are strongly associated ($\rho = 0.83$), while workflow presence has a weak positive association with stars ($\rho = 0.10$). The latter effect is small, but it is not evidence of statistical independence, nor does it identify why a maintainer adds a workflow.

Workflow presence and repository age have a negligible association ($\rho = -0.05$). The stronger negative association for recorded run count ($\rho = -0.19$) concerns volume rather than presence and may reflect exposure time or unmeasured repository differences. Consequently, the cross-sectional data do not establish a generational shift in workflow adoption.

The cross-topic analysis (Figures 9 and 11) shows descriptive variation in workflow presence and recorded activity. *Academic Annotation Workflow*, *AI-Powered Vault Assistant*, and *Comprehensive Task Management* have relatively high values on several measures, whereas *Canvas Productivity Toolkit* and *File Management Enhancements* have lower workflow-presence rates. The available metrics do not isolate whether complexity, external-service integration, contributor experience, topic size, or another factor explains these differences.

6. Threats to Validity

This section discusses potential threats to the validity of our study, organized along standard validity dimensions. We focus on the main protocol-level decisions that may affect the interpretation of our results.

6.1. Construct Validity

Construct validity concerns whether the study design adequately captures the constructs under investigation.

- **LLM-assisted taxonomy construction:** Our taxonomy relies on LLM-generated keyphrases and labels, whose final outputs were subsequently evaluated through independent manual annotation and third-annotator adjudication. LLM outputs may introduce biases related to prompt design and model behavior, potentially affecting how plugins are characterized and grouped. To mitigate this, we employed a structured prompting protocol targeting three complementary dimensions (problem, workflow, user context), enforced reproducibility through fixed generation parameters, and applied outlier filtering to discard degenerate outputs. The prompts and scripts are released in the replication package for scrutiny and replication. The manual assessment directly evaluates whether the generated keyphrases and resulting assignments are acceptable to human reviewers; however, it is not an ablation study and therefore does not establish that LLM enrichment outperforms raw metadata or a manually constructed taxonomy.

- **Absence of a gold-standard taxonomy:** No ground-truth classification exists for knowledge-centric plugin ecosystems, which limits our ability to validate the resulting clusters quantitatively. We mitigate this limitation through complementary evidence: manual assessment of a stratified 200-plugin sample, inspection of the hierarchy, recognizable functional domains, and the inter-topic similarity analysis. The manual design deliberately covers the 20 largest leaf topics but not all 60 topics, and its authors-as-annotators protocol does not constitute a community-derived gold standard. Validation by a larger panel of Obsidian users or plugin developers, particularly for smaller topics, would further strengthen construct validity. We therefore release all assignments and annotations in the replication package for independent examination.

- **Repository activity and automation operationalization:** We operationalized repository activity through PR, issue, commit, and star counts, active/inactive status through a one-year last-commit threshold, and automation through the presence and recorded runs of GitHub Actions workflows. Each proxy has a narrower meaning than the corresponding broader construct. A closed item need not indicate a merged PR or successfully resolved issue; an inactive repository may be stable rather than abandoned; a successful Actions run does not establish

code quality; and a workflow may automate tasks other than CI or deployment. We therefore report the indicators separately, include both absolute and per-plugin topic summaries, analyze workflow presence directly as a binary variable, and avoid treating their combination as proof of ecosystem health or engineering maturity.

- **Contributor demographics:** The mined repository metadata do not reveal contributors' education, occupation, professional status, experience, or motivations. We therefore cannot compare the developer population with professional-software-engineering communities or explain repository patterns through contributor background. References to a knowledge-centric ecosystem describe the platform and plugin functions, not a measured demographic composition.

6.2. Internal Validity

Internal validity concerns whether the observed patterns can be attributed to the variables under study rather than to confounding factors.

- **Cross-sectional study design:** All data were collected at a single point in time (February 10, 2026), providing a snapshot rather than a longitudinal view of the ecosystem. Plugins created shortly before the cutoff may appear to have low activity simply because they are new, while abandoned plugins may retain high cumulative metrics. To mitigate this, we distinguished between absolute and normalized metrics, and classified repositories as active or inactive based on their last commit date, rather than relying solely on cumulative counts.

- **Input data quality:** The quality, length, and language of README files vary substantially across plugins. Sparse or non-English READMEs may yield low-quality keyphrases, potentially distorting topic assignments. To mitigate this, we excluded plugins for which the LLM returned `NULL` (indicating insufficient or non-English content) and applied outlier filtering on keyphrase length. The document enrichment step—concatenating name, description, and keyphrases—ensured that even plugins with brief READMEs contributed meaningful input. In the manual sample, both primary annotators judged all 200 generated keyphrase sets to represent their corresponding plugins; the Wilson 95% confidence interval for the final acceptance proportion was 98.1%–100.0%. This supports the quality of the sampled enriched inputs, while not ruling out errors elsewhere in the corpus.

- **Model selection bias:** We used specific models (*gpt-oss-120b* for labeling, *Qwen3-Embedding-0.6B* for embeddings) whose linguistic or semantic biases may influence the resulting topic structure. Different LLMs or embedding models could produce different cluster compositions or label assignments. To mitigate this, we used open-weight models served locally for full reproducibility, and we provide all scripts and processed data in the replication package to enable replication with alternative models.

- **Confounding between topic size and maintenance metrics:** Larger topics mechanically produce higher aggregate PR and issue counts, which could create a spurious association between topic size and maintenance intensity. To mitigate this, we reported both totals and per-plugin means in Table 1, and the Spearman correlation analysis in RQ4 operates at the repository level rather than the topic level, controlling for this aggregation effect.

6.3. External Validity

External validity concerns the generalizability of our findings beyond the studied ecosystem.

- **Single-ecosystem scope:** Our study focuses exclusively on the Obsidian plugin ecosystem, which may limit generalizability to other knowledge-centric or hybrid communities (e.g., Notion, Logseq, Roam Research). Obsidian's specific architecture (local-first, Markdown-based, single community plugin registry on GitHub) may shape ecosystem dynamics in ways that do not transfer to platforms with different hosting models or governance structures. Obsidian is a valuable case of a plugin ecosystem serving knowledge-management workflows, but we do not claim that its contributor population is representative of developers, non-developers, or hybrid communities. A valid comparison with developer-centric ecosystems (e.g., VS Code or JetBrains) would require matched repository-selection criteria, observation windows, metric definitions, and contributor-demographic data.

- **GitHub observability and hosting bias:** The community registry links the studied plugins to public GitHub repositories, making commits, issues, PRs, stars, and Actions consistently observable. This design excludes private work and plugins distributed outside the registry, and privileges GitHub-native traces over development or support occurring elsewhere. Platforms centered on proprietary hosting, integrated marketplaces, or closed issue trackers would not expose equivalent measures; the reported workflow and contribution patterns should therefore not be transferred directly to Notion, Logseq, Roam Research, or other platforms.
- **Architecture and functional scope:** Obsidian is local-first and Markdown-based. Its file, note, link, and metadata abstractions may favor the prominence of file management, linking, publishing, and text-transformation topics in our taxonomy. Cloud-first or database-centered platforms, particularly those emphasizing synchronous collaboration or centrally hosted content, may yield a different functional structure even if they also support plugins. The taxonomy characterizes this platform's extension points rather than a universal taxonomy of knowledge tools.
- **Registry curation and population boundary:** Sampling the community-plugin registry gives the study a reproducible and practically complete boundary for listed plugins, but the registry is a curated population. Submission, inclusion, update, and removal practices can shape which repositories are visible, while experimental, rejected, unlisted, or independently distributed plugins are outside the frame. Registry presence must not be interpreted as evidence of plugin quality or sustained maintenance.
- **Technical contributor subgroup:** Creating and publishing an Obsidian plugin necessarily involves technical work, and GitHub-based development may select for contributors comfortable with public software-development infrastructure. However, we did not collect education, occupation, experience, or user-demographic data. The repository patterns characterize the observable plugin-producing subgroup and cannot be generalized to the broader Obsidian community or used to compare its technical profile with contributor populations on other platforms.
- **Ecosystem maturity and temporal context:** The Obsidian plugin ecosystem is relatively young (first plugins appeared in 2020). The activity split and workflow metrics are cross-sectional observations rather than stable long-term dynamics; in particular, the negligible association between workflow presence and repository age does not support a generational adoption claim. Longitudinal studies tracking these patterns over multiple years would clarify whether the observed trends persist or evolve.

7. Conclusion

This paper presented a comprehensive empirical study of the Obsidian plugin ecosystem through the lens of software engineering. By combining repository mining, semantic clustering, and LLM-assisted labeling on the full ecosystem of 2,675 community plugins, we addressed four research questions. Regarding the functional landscape (RQ1), our topic-modeling pipeline identified 60 leaf topics organized into six broad thematic families, with an ecosystem vocabulary centered on knowledge artifacts rather than programming constructs, and a notable emerging cluster of AI-powered plugins. Manual validation of 200 plugins from the 20 largest leaf topics yielded adjudicated acceptance rates of 100.0% for generated keyphrases, 95.5% for leaf-topic assignments, and 93.0% for parent-family assignments, supporting the content validity of the sampled major topics. Regarding the structural organization (RQ2), the inter-topic distance map, similarity matrix, and hierarchical clustering revealed a modular, long-tailed ecosystem with within-family semantic proximity, between-family separation, and cross-cutting semantic relationships in the model representations.

Regarding maintenance and evolution (RQ3), pull-request and issue data show recorded activity across all thematic families, with closed counts consistently exceeding open counts; these aggregate states do not establish resolution quality, responsiveness, or contributor demographics.

Regarding development practices (RQ4), 1,660 of 2,724 unique repositories (60.9%) contain a GitHub Actions workflow; among 1,068 with recorded runs, the median observed success proportion is 0.95. Stars and issues are strongly associated ($\rho = 0.83$), whereas workflow presence is only weakly associated with stars ($\rho = 0.10$) and negligibly associated with repository age ($\rho = -0.05$).

Our findings characterize repository-level structure, activity, and automation in a knowledge-oriented platform. They do not establish who the contributors are or how their backgrounds compare with those in developer-centric

ecosystems. More generally, the study is descriptive and observational: the reported associations do not identify causal mechanisms, and the absence of an equivalently sampled comparison ecosystem prevents claims that Obsidian is objectively similar to or different from developer-centric platforms.

More broadly, the results show that modern software ecosystems increasingly operate at the intersection of coding and thinking. As platforms like Obsidian grow, the boundary between development tools and knowledge-centric tools becomes more fluid, raising new questions for software engineering about how such ecosystems form, evolve, and are maintained.

Several avenues for future work emerge from this study. First, understanding who maintains these plugins—contributors from within the knowledge-centric community, external developers, or both—and the extent to which community sustainability depends on developer expertise would clarify the human dynamics behind the ecosystem. Second, longitudinal analyses tracking contributor diversity, release cadence, and the emergence of new thematic families (such as the AI cluster) over time would complement the cross-sectional snapshot provided here.

Third, a matched comparison with an ecosystem such as VS Code, JetBrains Marketplace, or the Chrome Web Store should apply equivalent registry-inclusion rules, observation dates, repository metrics, and taxonomy construction before testing whether the observed structural and activity patterns differ.

Acknowledgments

The study presented in this paper was funded in part by the Advise project, funded by the Spanish AEI with reference 2024/00416/002.

Data Availability

A complete reproduction package, including all scripts, processed data, and artifacts used in this study, is publicly available for inspection and reuse [28].

Declaration of Generative AI

During the preparation of this work, the authors used large language models in two capacities. First, as part of the research methodology, the open-weight model *gpt-oss-120b* was used to extract keyphrases from plugin README files and to generate topic labels, and the *Qwen3-Embedding-0.6B* model was used to compute document embeddings; these uses are fully described in Section 2 (Methodology). Second, during manuscript preparation, AI-assisted tools (*e.g., Grammarly*) were used to support drafting and editing of text. In both cases, the authors reviewed and edited all content as needed and take full responsibility for the content of the published article.

References

[1] Ahmed, T., Devanbu, P., Treude, C., Pradel, M., 2025. Can llms replace manual annotation of software engineering artifacts?, in: 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR), IEEE. p. 526–538. URL: http://dx.doi.org/10.1109/MSR66628.2025.00086, doi:10.1109/msr66628.2025.00086.

[2] Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., Teevan, J., Kikin-Gil, R., Horvitz, E., 2019. Guidelines for human-ai interaction, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. p. 1–13. URL: https://doi.org/10.1145/3290605.3300233, doi:10.1145/3290605.3300233.

[3] Baldwin, C., Clark, K., 2006. The architecture of participation: Does code architecture mitigate free riding? Management Science .

[4] Bosch, J., 2009. From software product lines to software ecosystems. Proceedings of the 13th International Software Product Line Conference .

[5] Campello, R.J., Moulavi, D., Sander, J., 2013. Density-based clustering based on hierarchical density estimates, in: Pacific-Asia conference on knowledge discovery and data mining, Springer. pp. 160–172.

[6] Crowston, K., Howison, J., 2006. The social structure of free and open source software development. First Monday .

[7] Crupi, G., Tufano, R., Velasco, A., Mastropaolo, A., Poshyvanyk, D., Bavota, G., 2025. On the effectiveness of llm-as-a-judge for code generation and summarization. IEEE Transactions on Software Engineering 51, 2329–2345. doi:10.1109/TSE.2025.3586082.

[8] Edirimannage, S., Elvitigala, C., Don, A.K.K., Daluwatta, W., Wijesekara, P., Khalil, I., 2024. Developers are victims too : A comprehensive analysis of the vs code extension ecosystem. URL: https://arxiv.org/abs/2411.07479, arXiv:2411.07479.

[9] Gonzalez-Barahona, J.M., 2024. Software development in the age of llms and xr, in: 2024 IEEE/ACM First IDE Workshop (IDE), pp. 66–69. doi:10.1145/3643796.3648457.

[10] Guglielmi, E., Rosa, G., Scalabrino, S., Bavota, G., Oliveto, R., 2022. Sorry, i don't understand: Improving voice user interface testing, in: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, pp. 1–12.
[11] Hsu, S., Tran, M., Fass, A., 2024. What is in the chrome web store?, in: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ACM. p. 785–798. URL: http://dx.doi.org/10.1145/3634737.3637636, doi:10.1145/3634737.3637636.
[12] Jansen, S., Finkelstein, A., Brinkkemper, S., 2009. A sense of community: A research agenda for software ecosystems, in: 2009 31st International Conference on Software Engineering-Companion Volume, IEEE. pp. 187–190.
[13] Ko, A.J., Myers, B.A., Aung, H.H., 2004. Six learning barriers in end-user programming systems, in: 2004 IEEE Symposium on Visual Languages-Human Centric Computing, IEEE. pp. 199–206.
[14] Lieberman, H., Paternò, F., Wulf, V., 2006. End user development. Springer.
[15] Lin, J., Sayagh, M., Hassan, A.E., 2023. The co-evolution of the wordpress platform and its plugins. ACM Trans. Softw. Eng. Methodol. 32. URL: https://doi.org/10.1145/3533700, doi:10.1145/3533700.
[16] Lucia, A.D., Fasano, F., Oliveto, R., Tortora, G., 2007. Recovering traceability links in software artifact management systems using information retrieval methods. ACM Transactions on Software Engineering and Methodology (TOSEM) 16, 13–es.
[17] MacCormack, A., Baldwin, C., Rusnak, J., 2012. Exploring the duality between product and organizational architectures: A test of the mirroring hypothesis. Research Policy .
[18] Marron, M., 2024. A new generation of intelligent development environments, in: Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments, Association for Computing Machinery, New York, NY, USA. p. 43–46. URL: https://doi.org/10.1145/3643796.3648452, doi:10.1145/3643796.3648452.
[19] Marshall, C., 2008. Rethinking personal digital archiving, part 1. D-Lib Magazine .
[20] McInnes, L., Healy, J., Melville, J., 2020. Umap: Uniform manifold approximation and projection for dimension reduction. URL: https://arxiv.org/abs/1802.03426, arXiv:1802.03426.
[21] Mockus, A., Fielding, R., Herbsleb, J., 2002. Two case studies of open source software development: Apache and mozilla. ACM Transactions on Software Engineering and Methodology 11, 309–346.
[22] Nardi, B., 1993. A small matter of programming: Perspectives on end user computing. MIT Press.
[23] Nghiem, K., Nguyen, A.M., Bui, N., 2024. Envisioning the next-generation ai coding assistants: Insights & proposals, in: Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments, Association for Computing Machinery, New York, NY, USA. p. 115–117. URL: https://doi.org/10.1145/3643796.3648467, doi:10.1145/3643796.3648467.
[24] Nguyen, P., Rocco, J., Di Sipio, C., Shakya, M., Di Ruscio, D., Di Penta, M., 2024. Automatic categorization of github actions with transformers and few-shot learning. doi:10.48550/arXiv.2407.16946.
[25] Pirolli, P., Card, S., 1999. Information foraging. Psychological review 106, 643.
[26] Qin, Z., Zhuang, H., Jagerman, R., Qian, X., Hu, P., Chen, C., Wang, X., Bendersky, M., Najork, M., 2021. Bootstrapping recommendations at chrome web store.
[27] Rosa, G., Moreno-Lumbreras, D., Kula, R.G., 2026a. Not only for developers: Exploring plugin maintenance for knowledge-centric communities, in: 2026 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE.
[28] Rosa, G., Moreno-Lumbreras, D., Kula, R.G., 2026b. Replication package. https://doi.org/10.6084/m9.figshare.31860265.
[29] Sas, C., Capiluppi, A., Sipio, C., Rocco, J., {Di Ruscio}, D., 2023. Gitranking: A ranking of github topics for software classification using active sampling. Software - Practice and Experience 53, 1982–2006. doi:10.1002/spe.3238. publisher Copyright: © 2023 The Authors. Software: Practice and Experience published by John Wiley & Sons Ltd.
[30] Sergeyuk, A., Titov, S., Izadi, M., 2024. In-ide human-ai experience in the era of large language models; a literature review, in: Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments, Association for Computing Machinery, New York, NY, USA. p. 95–100. URL: https://doi.org/10.1145/3643796.3648463, doi:10.1145/3643796.3648463.
[31] Spadini, D., Aniche, M., Bacchelli, A., 2018. Pydriller: Python framework for mining software repositories, in: Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering, pp. 908–911.
[32] Tiwana, A., 2014. Platform ecosystems: aligning architecture, governance, and strategy. Morgan Kaufmann.
[33] Zanartu, F., Treude, C., Cartaxo, B., Borges, H.S., Moura, P., Wagner, M., Pinto, G., 2022. Automatically categorising github repositories by application domain. URL: https://arxiv.org/abs/2208.00269, arXiv:2208.00269.