Evolution of Development Activity in NumPy: A Repository Mining Study

Cheikh Tidiane Mané

December 10, 2025

Abstract

This report investigates the evolution of development activity in the NumPy project using repository mining techniques. The study focuses on commit frequency, code churn, and long-term activity trends based on historical Git data.

1 Introduction

1.1 Context

NumPy is one of the core libraries of the Python scientific computing ecosystem. Understanding its development history can provide insight into how large open-source projects evolve over time.

1.2 Research Question

RQ1: How does development activity in the NumPy repository evolve over time?

1.3 Objectives

- Extract commit-level historical data using PyDriller.
- Analyze monthly commit trends and code churn.
- Identify long-term development patterns and major spikes.
- Provide a critical interpretation of the results.

2 Methodology

2.1 Tools and Frameworks

The analysis relies on PyDriller for mining Git history, Python for data processing, and Matplotlib/Seaborn for visualization.

2.2 Data Extraction

The following information was extracted from the NumPy repository:

- Commit hash
- Commit date
- Number of insertions and deletions
- Code churn (insertions + deletions)

Merge commits were excluded to focus on meaningful code changes. The data was aggregated at a monthly level.

2.3 Data Cleaning

The dataset was checked for missing values, duplicates, and correct date formatting. No missing values were found.

2.4 Reproducibility

All analysis steps can be reproduced by:

1. Installing dependencies using `requirements.txt`
2. Running the extraction script: `python3 src/extract_data.py`
3. Opening the notebook `analysis.ipynb` or running `python3 make_plots.py`

3 Results

3.1 Time Series of Monthly Commits

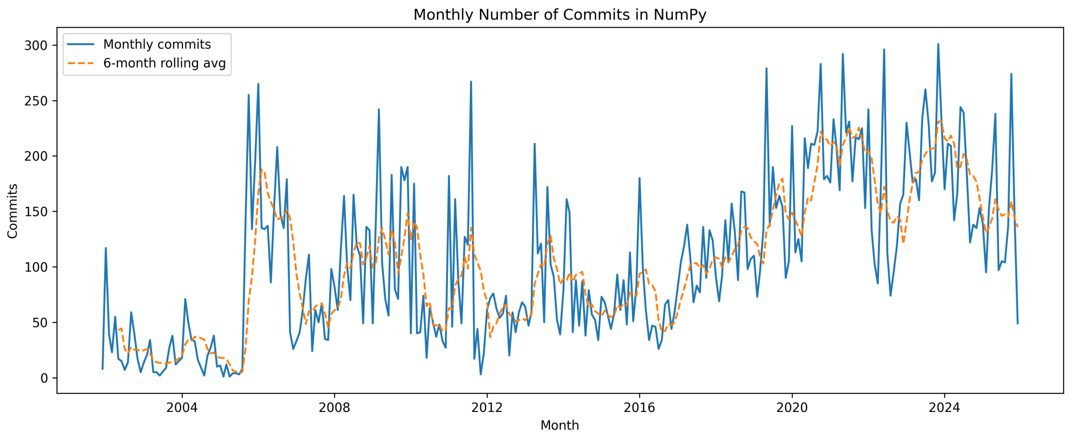

Figure 1: Monthly number of commits in the NumPy project.

3.2 Interpretation of the Monthly Commits Time Series

The time series plot exhibits substantial month-to-month variability in NumPy's development activity. Such short-term fluctuations are common in large open-source projects; however, the 6-month rolling average helps reveal several long-term patterns that characterize the evolution of the project.

Early phase (2001–2006): During the initial years of the project, monthly commit activity remains low and irregular. This period corresponds to the foundational stage of NumPy, when the codebase and contributor community were still being established. The rolling average stays close to zero, with only occasional small bursts of activity.

Growth and consolidation (2006–2012): Beginning around 2006, there is a clear rise in both monthly commit counts and the rolling average. This period marks the unification of Numeric and Numarray into NumPy and its increasing adoption across the scientific Python ecosystem. The frequent peaks in this interval reflect active feature development, API consolidation, and broader community involvement.

Mature stability (2012–2018): Commit activity remains highly variable month-to-month, but the rolling average tend to stabilize, indicating a mature and sustained development rhythm. This pattern is typical of a stable scientific library undergoing regular maintenance, bug fixes, and incremental enhancements.

Recent acceleration (2018–2024): In the later years of the timeline, development activity intensifies again, with some of the highest peaks observed in the entire dataset. The rolling average shows a distinct upward shift. This acceleration coincides with the rapid growth of machine learning and artificial intelligence: NumPy serves as the numerical foundation for major frameworks such as TensorFlow, PyTorch, JAX, and scikit-learn, and the increased demand for high-performance numerical computing has likely stimulated more contributions, optimizations, and refactoring efforts.

Overall trend: Despite high short-term volatility, the long-term trend shows a significant increase in development activity. NumPy has evolved from a small and irregularly updated project into a highly active and foundational component of the scientific and machine learning ecosystems.

3.3 Distribution of Monthly Code Churn

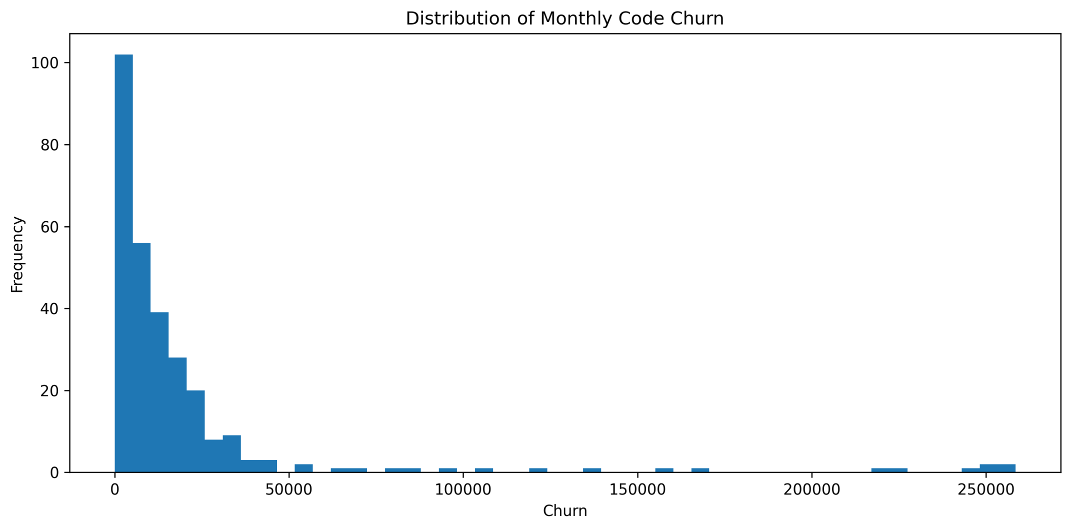

Figure 2: Distribution of monthly code churn (insertions + deletions).

3.4 Interpretation of the Code Churn Distribution

The distribution of monthly code churn is highly right-skewed, indicating that most months involve relatively modest levels of added and removed lines. These low-churn months correspond to routine maintenance work, such as incremental improvements, bug fixes, and small-scale refactoring. The dense concentration of values near the lower end of the distribution suggests that NumPy's development process is typically dominated by frequent but manageable updates.

In contrast, the histogram exhibits a long tail of extreme churn values. These occasional high-churn months reflect large-scale modification events, such as major refactoring, restructuring of core components, or substantial feature introductions. Such episodes are expected in a long-lived scientific computing library and often coincide with significant architectural transitions or ecosystem changes.

The combination of many small updates and few large restructuring phases highlights the heterogeneous nature of NumPy's evolution: long periods of incremental maintenance punctuated by intensive bursts of development activity. Some of these high-churn periods likely align with broader technological shifts, including the growing dependence on NumPy within modern machine learning and AI frameworks, which has increased the demand for performance and API modernization.

3.5 Year-Month Heatmap of Activity

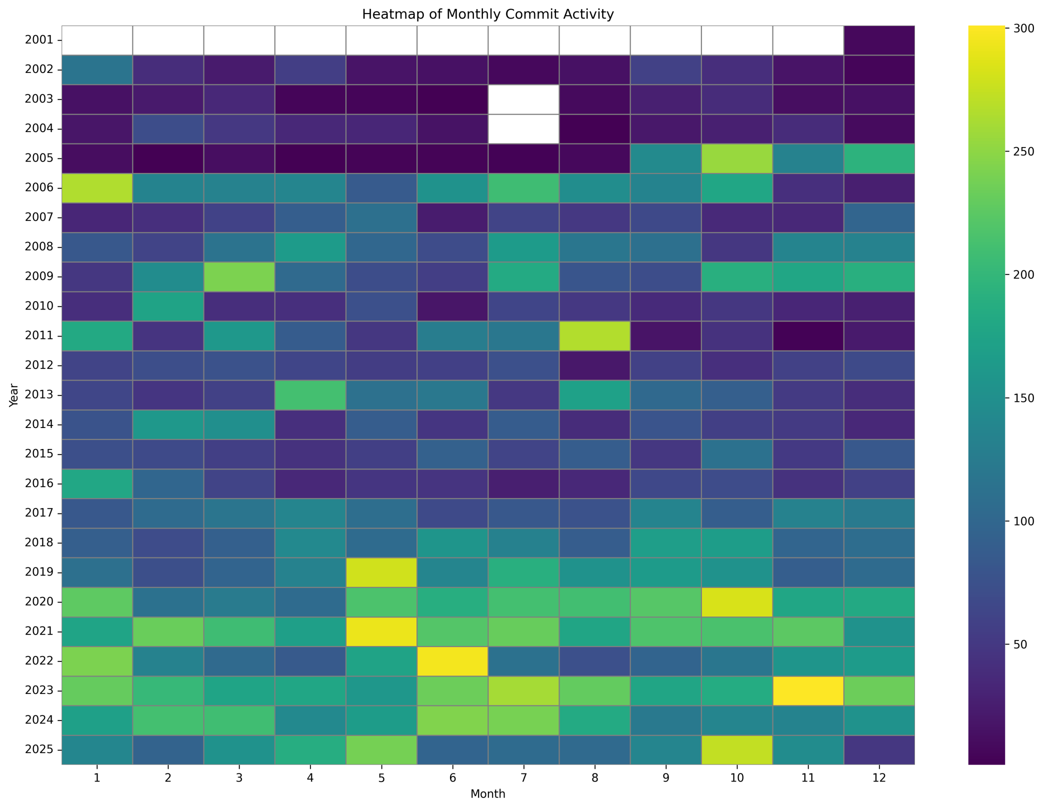

Figure 3: Heatmap of monthly commit activity by year.

3.6 Interpretation of the Year-Month Commit Heatmap

The heatmap provides a two-dimensional view of NumPy's development activity across its entire history, showing how the number of commits varies by month within each year. The color intensity highlights both short-term patterns (within a single year) and long-term trends (across multiple years).

Early years (2001–2005). The top of the heatmap shows mostly dark cells, indicating very low activity in the earliest years. This pattern is consistent with a young project still in its formative stages.

Increasing activity and expansion (2006-2009). Starting around 2006, the heatmap becomes noticeably brighter and more populated. Commits occur in more months, and activity is more evenly distributed throughout the year. This shift corresponds to the consolidation of NumPy as the primary numerical computing library in Python and the expansion of its contributor base. Although few months remain less active, the overall color pattern indicates clear growth in sustained development.

Stable but fluctuating mid-period (2010-2017). The middle section of the heatmap shows a more consistent baseline of commit activity. Most months exhibit medium-intensity colors, suggesting steady and ongoing development work. However, occasional brighter cells indicate short bursts of higher activity. This pattern reflects the maturity of the project: regular maintenance, continuous improvements, and predictable development cycles.

High-activity era (2018-2025). In the most recent years, the heatmap displays several bright yellow and green cells, representing some of the highest monthly commit counts in the project's history. Activity is sustained across nearly all months with fewer low-activity periods. This intensified development is likely linked to broader ecosystem demands, including increased reliance on NumPy in machine learning and AI frameworks such as TensorFlow, PyTorch, and JAX. These external pressures have driven modernization, performance enhancements, and ongoing refactoring efforts.

Seasonal patterns. Across all years, the heatmap does not reveal a strong or consistent seasonal pattern. There is no clear tendency for particular months to be systematically more or less active.

General trend. The progression from dark colors at the top of the heatmap to brighter illustrates NumPy's long-term evolution from a small, irregularly maintained project into a highly active and essential component of the scientific Python ecosystem.

3.7 Commits–Churn Scatter Plot

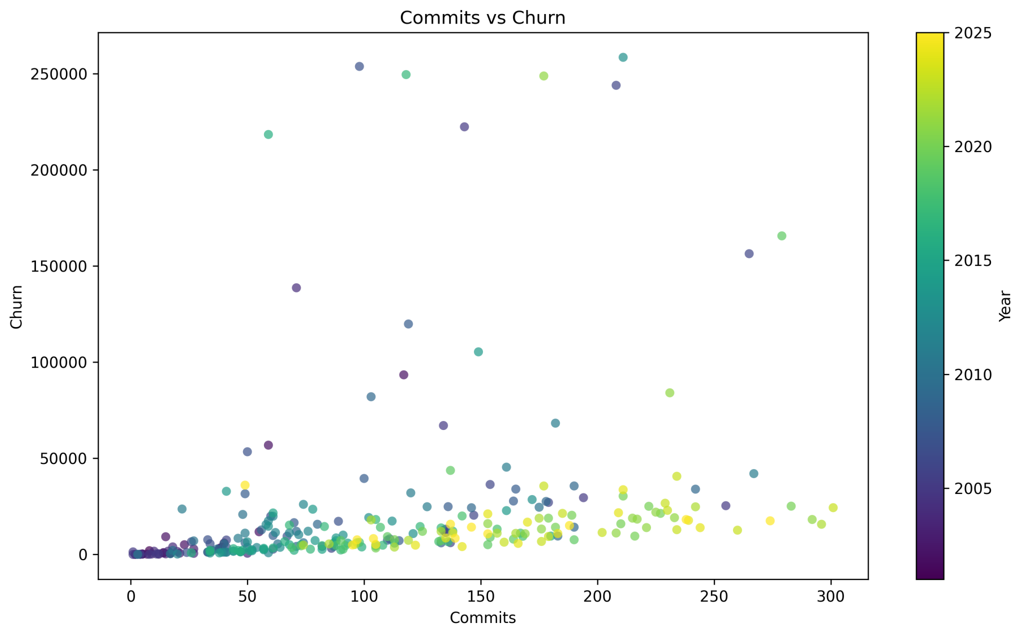

Figure 4: Monthly Commits and Code Churn.

3.8 Interpretation of the Commits–Churn Scatter Plot

The scatter plot illustrates the relationship between the number of monthly commits and the corresponding code churn with each point colored according to the year in which the activity occurred. We can see that :

Low churn dominates most months : A dense cluster of points appears near the bottom of the plot, indicating that the majority of months exhibit very low churn regardless of the number of commits. This suggests that much of NumPy's development consists of small, incremental changes where individual commits introduce limited modifications. Consequently, commit frequency alone is not a strong indicator of the magnitude of code change.

Presence of high-churn outliers : A small number of points lie much higher on the churn axis, representing months with exceptionally large volumes of added or removed code. These outliers may reflect major refactoring efforts, significant feature introductions, or large-scale restructuring of the codebase.

Temporal evolution visible through the color gradient : The color mapping highlights a temporal trend: more recent years tend to show a broader spread of commit activity and moderate churn levels. This pattern aligns with the increasing importance of NumPy within the scientific computing and machine learning ecosystems. High-churn events, however, appear across multiple periods, indicating that large-scale modifications have occurred throughout the project's history.

Weak correlation between commits and churn : The absence of a clear upward trend in the scatter indicates that commit count is not strongly correlated with churn. Months with many commits can still exhibit low churn, while some months with few commits produce substantial modifications. This reflects heterogeneous development practices, where some contributors make many small commits and others perform fewer but larger changes.

Overall interpretation : Taken together, the plot highlights NumPy's development dynamics: predominantly incremental month-to-month work, punctuated by occasional bursts of intensive, high-impact code modifications. It also shows that churn is a more meaningful indicator of structural evolution than raw commit frequency.

4 Discussion

4.1 Key Findings

The visual analyses reveal several important patterns in the evolution of NumPy's development activity:

- **Long-term growth in activity:** The monthly time series shows a clear increase in commit activity over the project's lifetime, progressing from early irregular contributions to sustained and higher levels of development in recent years.
- **Small incremental changes dominate:** The distribution of monthly code churn is highly right-skewed, indicating that most months involve modest modifications, while a few months exhibit extremely large-scale changes such as refactoring or restructuring.
- **Weak relationship between commits and churn:** The commits-churn scatter plot shows no strong correlation between commit frequency and the magnitude of code change. High-commit months often produce little churn, while some low-commit months involve substantial modifications.
- **Overall evolution:** Together, the results portray NumPy as a project that has evolved from a small, irregularly maintained codebase into a highly active and continuously developing scientific software foundation.

4.2 Explanations

Several factors help explain the observed patterns in NumPy's development activity:

- **Early low activity** reflects the project's formative stage, with few contributors and an evolving architecture.
- **Increased activity around 2006** corresponds to NumPy's adoption as the core numerical library in Python, following the unification of Numeric and Numarray and leading to broader community.
- **Large churn spikes** observed in the analysis indicate months with unusually extensive code modifications. While the exact causes cannot be determined from commit metadata alone, such spikes are typically associated with majorre factoring, modernization efforts, or structural updates in long-lived scientific software projects.
- **Recent sustained activity** aligns with the growing reliance on NumPy within machine learning and AI frameworks, which drives continued performance and compatibility improvements.
- **Absence of seasonal effects** suggests that development is shaped by contributor availability and ecosystem transitions rather than calendar cycles.

5 Validity and Limitations

While the analysis provides meaningful insights into NumPy's evolution, several limitations should be acknowledged.

- **Commit-based metrics are imperfect proxies:** Commit counts and churn do not directly reflect development effort or code quality. Many small commits may represent limited work, whereas a single commit may introduce substantial complexity.
- **Extraction limitations:** The results depend on PyDriller's extraction of Git metadata and I did not include rebasing practices or historical restructuring of the repository.
- **Lack of contextual information:** The study does not incorporate release schedules, contributor discussions, or external ecosystem events, limiting the ability to associate activity patterns with specific project decisions.
- **Aggregation effects:** Monthly aggregation smooths short-term fluctuations and may obscure fine-grained development behaviors that occur at the daily or weekly level.

These limitations do not undermine the overall results but highlight the need for cautious interpretation and suggest that commit-based analyses should be complemented with additional qualitative or contextual data when available.

6 Future Work

Several directions could extend this study. First, future analyses could examine contributor behavior more closely by tracking the number of active developers over time and assessing how contributor turnover influences development activity. Second, a file or module-level investigation could identify which components of the codebase are responsible for the largest spikes in churn and how different architectural areas evolve.

Third, correlating commit activity with NumPy's release cycle or major ecosystem events such as Python version transitions or the growth of machine learning frameworks would provide additional context for interpreting the observed patterns. Finally, incorporating qualitative data such as issue discussions, pull request histories, or governance decisions could offer deeper insight into the motivations behind intensive development periods or large-scale refactoring efforts.

7 AI Usage Transparency

ChatGPT was used only to correct linguistic errors, resolve minor LaTeX syntax issues, and provide occasional assistance with debugging. All ideas, reasoning, methodological decisions, and interpretations of the results originate from me. I also performed the data extraction and data analysis.

8 Conclusion

This study examined the long-term evolution of development activity in the NumPy project through commit metadata, code churn metrics, and temporal visualizations. The results reveal a progression from sparse and irregular early contributions to sustained and increasingly intense development in recent years. While most months are characterized by small incremental updates, several periods of exceptionally high churn correspond to major structural changes within the project.

The heatmap and scatter plot highlight how development activity has broadened over time and show that code churn does not correlate directly with commit frequency. Together, these findings illustrate how NumPy has evolved into a highly active and continuously modernized scientific software foundation. The recent acceleration in activity aligns with broader ecosystem trends, particularly the growing reliance on NumPy within machine learning and AI frameworks.

While the analysis provides meaningful insights, it also underscores the need to complement commit-based metrics with contextual and qualitative information. Future work may expand this study by investigating contributor dynamics, module-level evolution, and the relationship between development activity and major external events.