The Sampling Threat when Mining Generalizable Inter-Library Usage Patterns

Yunior Pacheco Correa[1], Coen De Roover[1] and Johannes Härtel[2]

[1]Vrije Universiteit Brussel, Brussels, Belgium
[2]Vrije Universiteit Amsterdam, Amsterdam, Netherlands

Abstract

Tool support in software engineering often relies on relationships, regularities, patterns, or rules mined from other users' code. Examples include approaches to bug prediction, code recommendation, and code autocompletion. Mining is typically performed on samples of code rather than the entirety of available software projects. While sampling is crucial for scaling data analysis, it can affect the generalization of the mined patterns.

We observe that limiting the sample to a specific library may hinder the generalization of inter-library patterns, posing a threat to their use or interpretation. Using a simulation and a real case study, we show this threat for different sampling methods. Our simulation shows that only when sampling for the disjunction of both libraries involved in the implication of a pattern, the implication generalizes well. Additionally, we show that real empirical data sampled using the GitHub search API does not behave as expected from our simulation. This identifies a potential threat relevant for many studies that use the GitHub search API for studying inter-library patterns.

Keywords

Sampling, Usage Patterns, Inter-Library, Dataset, Data Mining

1. Introduction

Sampling is crucial in empirical research, including Empirical Software Engineering (ESE) and Mining Software Repositories (MSR) [1, 2]. In MSR and ESE, researchers often sample software projects from sources like GitHub and aim to generalize their findings to unseen software projects. For studies that mine library or framework (API) patterns, sampling is equally important. Researchers extract API usage patterns from code by sampling examples from existing API applications.

Nuryyev et al. [3] for instance, mined the sample of 533 repositories from GitHub for annotation usage rules and validated them by human experts. We can find a rule of the following form:

$$type(\text{javax.json.JsonString}) \rightarrow annotation(\text{org.eclipse.microprofile.jwt.Claim})$$

This rule can be interpreted as a logical or probabilistic relationship, indicating that when a method returns a JsonString, it should carry a Claim annotation. This statement crosses different libraries, making it an inter-library pattern. Since JavaX's JsonString may also appear in other contexts, unrelated to MicroProfile's Claim annotation, the rule is not logically true. However, it may hold probabilistically with a certain confidence.

Nuryyev et al. discovered this pattern with unexpectedly high confidence. The authors assume JavaX to be an integral part of the MicroProfile framework. This is not true and renders it a rule between different libraries. We call it an *inter-library pattern* instead of an *intra-library pattern*. From a sampling perspective, we noticed that this distinction can be crucial. This problem leads us to ask: Is the confidence of a rule computed on the sample the same as for the entire population? In short, can we generalize? Is there a difference between intra-library and inter-library patterns? More concretely, we define our **research question** as follows:
How do sampling methods influence the generalizability of mined inter-library usage patterns from client software projects?

The 24th Belgium-Netherlands Software Evolution Workshop 17–18 November 2025, Enschede, The Netherlands
ypacheco@vub.be (Y. P. Correa); coen.de.roover@vub.be (C. D. Roover); j.a.hartel@vu.nl (J. Härtel)
0000-0002-7849-7841 (Y. P. Correa); 0000-0002-1710-1268 (C. D. Roover); 0000-0002-7461-2320 (J. Härtel)


2. Research Method and Results

To answer the research question, this paper follows a *research method* that combines an *empirical study* and a *simulation study* that examines inter-library patterns mined on data using different practices: i) *random samples*, ii) *single library samples* that collect client projects that use a particular library, and iii) *co-used library samples* that collect client projects by analyzing different combinations of usage of two libraries.

In the simulation study, we analyze the *generalizability* in terms of *confidence* for both intra-library and inter-library patterns. In the empirical study, we focus on inter-library patterns. The empirical and simulation studies examine the impact of different sampling methods on the mined inter-library patterns, considering the degree of popularity of the libraries involved.

We present evidence that *confidence* of mined inter-library usage patterns differs depending on the sampling method. Thereby, patterns may or may not generalize. For the specific case of mining inter-library usage patterns, we present an empirical study showing that mining on data sampled from GitHub does not behave as expected based on a corresponding simulation. Specifically, two sampling methods that should theoretically yield the same results instead produce different outcomes. This discrepancy suggests that at least one of the sampling methods is unreliable, assuming our simulation assumptions hold. Our findings highlight that the GitHub search API operates more as a black box than previously anticipated.

We provide a replication package online[1]. The resulting insights of our study can directly be used by future studies that extract patterns from samples. Either they **improve their sampling method in a way that the mined patterns generalize better**, or they **improve their threats to validity section by discussing the limitations that we identified**.

3. Conclusions and Future Work

At its core, this study examines the generalizability of findings in a specific area of software engineering. Our insights have practical implications: they can inform the choice of sampling methods in future research, or be highlighted as potential threats to validity in similar studies.

Relying only on data from single-library sampling for inter-library patterns (without manual validation) can weaken the validity of results. Random sampling is better but often not practical for rare libraries. Combining data from multiple sources and considering alternative mirror datasets like GHTorrent is recommended when possible.

Researchers and practitioners must stay alert. Recognizing and documenting the limitations of data collection and the black box nature of the GitHub search API is important to ensure the validity of mining studies. Developers using mined usage patterns in tools should also be careful, especially when working with patterns involving less popular libraries.

Future work should conduct more experiments to confirm these findings in similar settings. It is important to address and resolve the issues identified in this study. Further research should develop methods that help ensure the generalizability of results when mining inter-library usage patterns. This includes providing more transparent indexing mechanisms as alternatives to GitHub.

References

[1] V. Cosentino, J. L. C. Izquierdo, J. Cabot, Findings from github: methods, datasets and limitations, in: MSR, ACM, 2016, pp. 137–141.

[2] O. Dabic, E. Aghajani, G. Bavota, Sampling projects in github for MSR studies, in: MSR, IEEE, 2021, pp. 560–564.

[3] B. Nuryyev, A. K. Jha, S. Nadi, Y. Chang, E. Jiang, V. Sundaresan, Mining Annotation Usage Rules: A Case Study with MicroProfile, in: ICSME, IEEE, 2022, pp. 553–562.

[1] https://doi.org/10.5281/zenodo.14841462