Analyzing the Impact of Documentation on Performance Metrics in Different Continuous Integration Open-Source Projects

Author: Daniel Rachev (d.n.rachev@student.tudelft.nl)
Supervisors: Sebastian Proksch, Shujun Huang

TU Delft

Introduction

Continuous Integration (CI) is a standard practice, but the impact of documentation on its performance is poorly understood. This study quantitatively investigates the link between documentation practices and key DevOps metrics (KPIs) in 670 open-source projects.

Research Questions

RQ1: Is documentation completeness correlated with delivery frequency?

RQ2: Does documentation update frequency correlate with defect count?

RQ3: Are documentation changes in release cycles correlated with mean time to recovery for reported issues?

Methodology

Data Selection

We gathered 670 CI projects following a set of criteria:

- Python, JavaScript, TypeScript, Java, C#, C++, PHP
- Not a fork / archived project
- > 50 stars, ≥ 50 commits, ≥ 20 releases
- ≥ 100 issues, and ≥ 75% of them labeled

Metrics

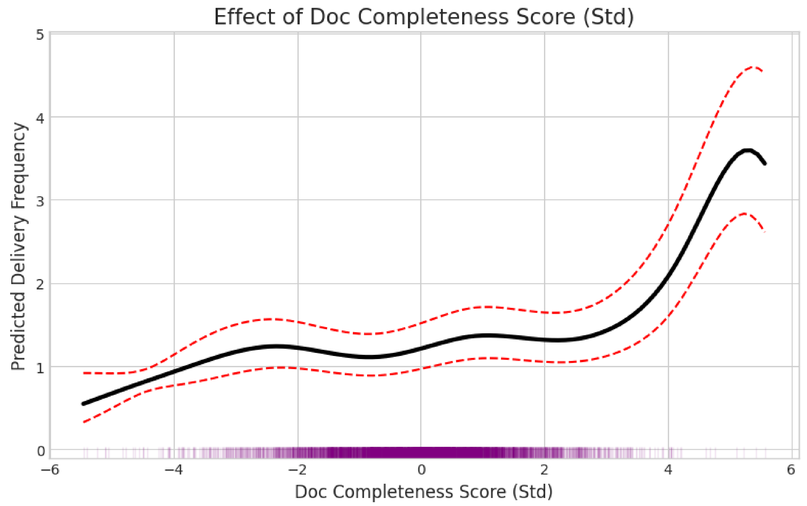
Metric	How We Measured It
Completeness Score	Composite of standard files, their length, & comment volume
Update Ratio	% of commits in 30 days changing docs
Release Size	# of release note lines per 30 days
Delivery Freq.	# of official releases per 30 days
Defect Count	# of open "bug" issues per 30 days
MTTR	Avg. time to close "bug" issues

Analysis

We analyzed 12 months of data using Generalized Additive Mixed Models (GAMMs) to model non-linear relationships.

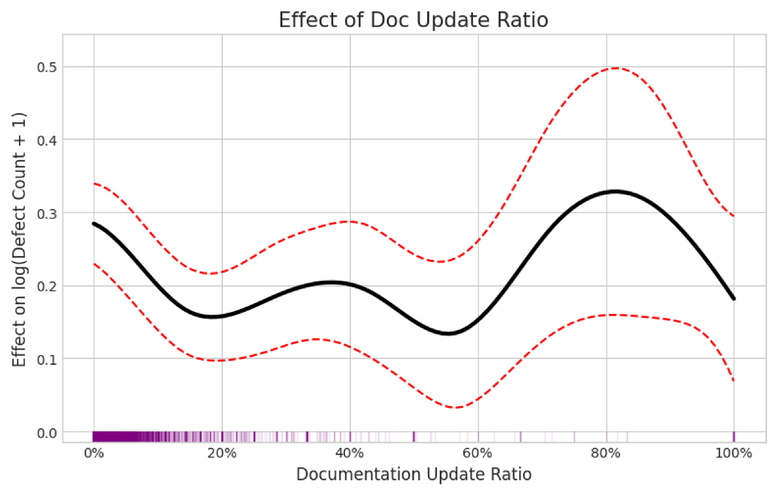
Results

RQ1: Documentation completeness has a "tipping point" for delivery frequency.



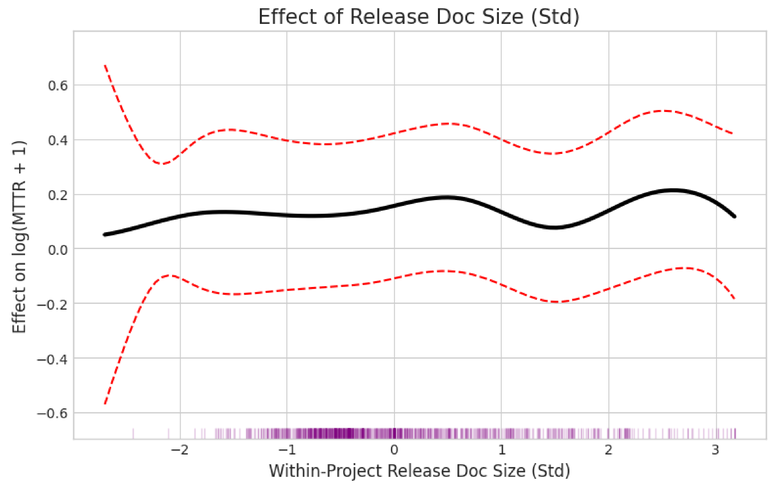
Delivery frequency increases dramatically when documentation completeness score exceeds +3.0, indicating a critical mass effect.

RQ2: A "sweet spot" exists for documentation update frequency.



Defect counts are lowest when 20-55% of commits update documentation, suggesting a healthy project rhythm.

RQ3: The volume of release notes does not impact recovery time.



No significant correlation was found between release note size and MTTR, suggesting length is not a proxy for usefulness in incident recovery.

Conclusions

Key Takeaways for Practitioners

- **Invest in Excellence:** Basic docs are helpful, but achieving a high standard can accelerate delivery speed.
- **Doc Update Ratio as a Health Metric:** If the ratio is too low (<20%), you're accumulating technical debt. Too high (>55%) may signal disruptive refactoring.
- **Technical Docs > Long Release Notes:** When an issue occurs, developers need clear, up-to-date technical documentation to solve it. Focus effort where it has the most impact.

Limitations & Future Work

- **Limitations:** Bug identification is keyword-based; documentation analysis is limited to the repository itself (e.g., no external wikis).
- **Future Work:** NLP to analyze documentation content quality; apply causal inference models.