Understanding the Helpfulness of Stale Bot for Pull-based Development

An Empirical Study of 20 Large Open-Source Projects

SAYEDHASSAN KHATOONABADI, Concordia University, Canada
DIEGO ELIAS COSTA, Concordia University, Canada
SUHAIB MUJAHID, Mozilla Corporation, Canada
EMAD SHIHAB, Concordia University, Canada

Pull Requests (PRs) that are neither progressed nor resolved clutter the list of PRs, making it difficult for the maintainers to manage and prioritize unresolved PRs. To automatically track, follow up, and close such inactive PRs, Stale bot was introduced by GitHub. Despite its increasing adoption, there are ongoing debates on whether using Stale bot alleviates or exacerbates the problem of inactive PRs. To better understand if and how Stale bot helps projects in their pull-based development workflow, we perform an empirical study of 20 large and popular open-source projects. We find that Stale bot can help deal with a backlog of unresolved PRs as the projects closed more PRs within the first few months of adoption. Moreover, Stale bot can help improve the efficiency of the PR review process as the projects reviewed PRs that ended up merged and resolved PRs that ended up closed faster after the adoption. However, Stale bot can also negatively affect the contributors as the projects experienced a considerable decrease in their number of active contributors after the adoption. Therefore, relying solely on Stale bot to deal with inactive PRs may lead to decreased community engagement and an increased probability of contributor abandonment.

CCS Concepts: • **Human-centered computing → Empirical studies in collaborative and social computing**; **Open source software**; • **Software and its engineering → Collaboration in software development**; **Open source model**.

Additional Key Words and Phrases: Software development bots, pull request abandonment, pull-based development, modern code review, social coding platforms, open-source software

ACM Reference Format:
SayedHassan Khatoonabadi, Diego Elias Costa, Suhaib Mujahid, and Emad Shihab. 2023. Understanding the Helpfulness of Stale Bot for Pull-based Development: An Empirical Study of 20 Large Open-Source Projects. *ACM Trans. Softw. Eng. Methodol.* 1, 1 (September 2023), 43 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: SayedHassan Khatoonabadi, Data-driven Analysis of Software (DAS) Lab, Department of Computer Science & Software Engineering, Concordia University, Montreal, QC, Canada, sayedhassan.khatoonabadi@mail.concordia.ca; Diego Elias Costa, Department of Computer Science & Software Engineering, Concordia University, Montreal, QC, Canada, diego.costa@concordia.ca; Suhaib Mujahid, Mozilla Corporation, Montreal, QC, Canada, smujahid@mozilla.com; Emad Shihab, Data-driven Analysis of Software (DAS) Lab, Department of Computer Science & Software Engineering, Concordia University, Montreal, QC, Canada, emad.shihab@concordia.ca.

1049-331X/2023/9-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Open-source projects widely adopt pull-based development as a more efficient and effective alternative to traditional methods for contributing and reviewing code changes [19, 75]. In this development model, contributors submit a Pull Request (PR) to suggest changes for integration into the project. The PR is then reviewed by the project maintainers and updated by the contributor until it is ready to be merged. However, the review process of some PRs is left unfinished due to either the contributor not addressing the maintainers' comments or the maintainers not following up on the progress of the PR [30, 37, 59]. If neither progressed nor resolved, such inactive PRs accumulate over time, clutter the list of PRs, and eventually make it difficult for the maintainers to manage and prioritize unresolved PRs [21, 37]. As a real-world example, a large backlog of unresolved PRs led the DefinitelyTyped project to declare "bankruptcy" in June 2016. Consequently, they closed all unresolved PRs submitted before May 2016 just to be able to start afresh [55].

Manually keeping track of inactive PRs, following up on their progress, and closing them if needed places an additional burden on the project maintainers who are already occupied with other development tasks [21, 30, 37]. To free the maintainers from manually triaging such PRs, Stale bot [15] was released in 2017 and since then has been increasingly adopted by open-source projects on GitHub[1]. As shown in Figure 1, Stale bot aims to make the status of open and not progressing PRs explicit by automatically labeling, commenting, and closing PRs after a pre-configured period of inactivity [29]. Nevertheless, there are ongoing debates within the open-source community on whether using Stale bot alleviates or exacerbates the problem of inactive PRs. The creators of Stale bot claim that based on the experience of hundreds of projects and organizations, Stale bot is an effective method for focusing on the work that matters most [29]. Conversely, part of the community regards Stale bot as "harmful" [6] and a "false economy" [68]. They argue that while Stale bot may initially seem helpful, it results in duplicated PRs, fragmented information, and eventually frustration in the community. Some studies have also incidentally mentioned that Stale bot can introduce noise and friction for both the contributors and the maintainers [7, 38, 47, 63, 66].

Despite all these positive and negative claims, the helpfulness of Stale bot has not yet been empirically validated. Therefore, we set out to better understand if and how adopting Stale bot helps open-source projects in their pull-based development workflow. This investigation is particularly important as Stale bot is commonly used to deal with inactive or abandoned PRs. Towards this goal, we perform an empirical study [5, 33] of 20 large and popular open-source projects on GitHub that have used Stale bot for at least one consecutive year. Specifically, we aim to answer the following research questions in this paper:

RQ_1: How much do the studied projects use Stale bot to deal with their PR backlog? We analyze the configuration and activity of Stale bot to understand the extent to which large open-source projects rely on Stale bot to automatically deal with their unresolved PRs. Our results show that the usage level of Stale bot widely varies among the studied projects. On average each month, Stale bot intervened in less than 25% of open PRs in nine projects, between 25% and 50% of open PRs in five projects, and more than 50% of open PRs in six projects. Unexpectedly, the projects with a larger backlog of unresolved PRs have typically not relied more aggressively on Stale bot to deal with their unresolved PRs.

RQ_2: What is the impact of Stale bot on pull-based development in the studied projects? We apply interrupted time-series analysis [58] as a well-established quasi-experiment to understand if and how adopting Stale bot improves the efficiency and effectiveness of the pull-based development workflow in large open-source projects. Our results show that the

[1]We observed that on average 7% more projects had adopted Stale bot each month till October 2021.

Fig. 1. An example prompt by Stale bot

studied projects closed more PRs within the first few months of adopting Stale bot, but overall closed and merged fewer PRs afterward. The adoption of Stale bot is also associated with faster first reviews in merged PRs, faster resolutions in closed PRs, slightly fewer updates in merged PRs, and considerably fewer active contributors in the projects.

RQ_3: What kind of PRs are usually intervened by Stale bot in the studied projects? We analyze the characteristics of PRs intervened by Stale bot, as well as their contributors and review processes, to understand the factors that are associated with a higher probability of getting intervened by Stale bot in large open-source projects. Our results show that Stale bot tends to intervene more in complex PRs, PRs from novice contributors, and PRs with lengthy review processes. Specifically, besides the resolution time of PRs, the largest differences are observed in the number of prior PRs by contributors, the mean response latency of PRs, the acceptance rate of contributors, and the contribution period of contributors.

Our findings imply that adopting Stale bot helped the studied projects deal with their accumulated backlog of unresolved PRs. Stale bot has also improved the efficiency of the review process of PRs by helping the maintainers focus on PRs that are more likely to get merged. Despite these advantages, the adoption of Stale bot also brings some disadvantages. For example, the projects experienced a decrease in their number of active contributors after the adoption. Besides, Stale bot also tends to intervene more in PRs submitted by novice contributors. However, such contributors are the ones who face the most barriers and thus need the most guidance from the maintainers [51–53]. Prior studies have also highlighted the importance of attracting newcomers to ensure the sustainability of projects [74]. Therefore, relying solely on Stale bot to deal with inactive PRs may lead to decreased community engagement and an increased probability of contributor abandonment. In conclusion, our study provides a better understanding of the potential benefits and drawbacks of employing Stale bot within a pull-based development workflow.

Our Contributions. In summary, we make the following contributions in this paper:

- To the best of our knowledge, this is the first in-depth study investigating the helpfulness of Stale bot for the pull-based development workflow in large open-source projects.
- We provide empirical evidence on the reliance of projects on Stale bot to deal with their PR backlog, the impact of Stale bot on pull-based development, and the kind of PRs usually intervened by Stale bot.
- To promote the reproducibility of our study and facilitate future research on this topic, we publicly share our dataset online at https://doi.org/10.5281/zenodo.7978381.

Paper Organization. The rest of the paper is organized as follows. Section 2 reviews the related work and Section 3 overviews the dataset used for our study. Then, Sections 4 to 6 report our approach and findings to answer each research question, and Section 7 discusses the implications of our study. Finally, Section 8 describes the limitations of our study, and Section 9 concludes this paper with a summary of our study.

2 RELATED WORK

In the following, we first overview research regarding the usage and challenges of Stale bot before moving on to studies on the reasons and consequences of PR abandonment. Next, we review studies on the impact of tools on pull-based development and then summarize studies on the review latency and decision of PRs.

Usage and Challenges of Stale Bot. Stale bot was released in 2017 to automatically triage abandoned issues and PRs and is adopted by many open-source projects on GitHub [15, 29, 64]. However, little is known about the usage and challenges of Stale bot in the literature. In a preliminary study, Wessel et al. [64] investigated how projects adapt and maintain Stale bot over time by analyzing its configuration history in 765 open-source projects. They found that most projects use Stale bot to triage both their issues and PRs. Furthermore, they found that while most projects do not modify the configuration of Stale bot after its initial setup, the few that do rarely make more than three modifications subsequently. Therefore, they concluded that setting up and using Stale bot does not require much effort from the projects. Several studies [7, 38, 47, 63, 66] have also incidentally mentioned that Stale bot introduces noise and friction for both the contributors and the maintainers.

Nevertheless, if and how Stale bot can actually be helpful to open-source projects has not yet been studied. To fill this knowledge gap, our paper empirically studies the helpfulness of Stale bot in the context of pull-based development. Specifically, we investigate the reliance of projects on Stale bot to deal with their PR backlog, the impact of Stale bot on pull-based development, and the kind of PRs usually intervened by Stale bot.

Reasons and Consequences of PR Abandonment. PR abandonment as a challenge that wastes time and effort of both contributors and maintainers has only recently received attention from the literature. Li et al. [37] attempted to understand the reasons for, the impacts of, and the coping strategies for abandoned PRs by qualitatively studying five popular open-source projects on GitHub. They observed that PRs are abandoned mostly due to the lack of maintainers' responsiveness and the lack of contributors' time and interest. They also reported that PR abandonment increases the efforts needed to manage and maintain projects because abandoned PRs clutter the list of PRs, waste review efforts, require additional attention for a careful closing, delay the landing of interdependent PRs, result in duplicated PRs, disorder project milestones, and finally leave a bad impression.

As a complementary study, Khatoonabadi et al. [30] conducted a mixed-methods study using both quantitative and qualitative methods to better understand the underlying dynamics of PRs abandoned by their contributors in 10 popular and large open-source projects on GitHub. By investigating the influence of various factors related to PRs, contributors, review processes, and projects on the abandonment probability of PRs, they found that complex PRs, PRs from novice contributors, and PRs with long discussions are more likely to get abandoned. They also observed that the most frequent reasons for PR abandonment are related to the obstacles faced by the contributors, followed by the hurdles imposed by the maintainers during the review process of PRs. While these studies investigated the reasons and consequences of PR abandonment, our study focuses on the usage and impact of Stale bot as a common strategy to deal with abandoned PRs in open-source projects.

Impact of Tools on Pull-based Development. Open-source projects are adopting various automation tools and bots to improve the efficiency and effectiveness of their pull-based development workflow [54, 60, 61]. Several studies have evaluated the impact of adopting such tools in open-source projects by applying interrupted time-series analysis [58] (a.k.a. regression discontinuity design). Zhao et al. [73] was the first to apply this method for understanding the impact of adopting Travis CI. They found that the adoption slows down the increasing trend in the number of merge commits, accelerates the decreasing trend in the merge commit churn, and reverses the increasing trend in the number of closed PRs. Cassee et al. [2] further studied the impact of adopting Travis CI and found that it also slows down the increasing trend in the number of discussion comments in PRs.

Wessel et al. [62] studied the impact of adopting code review bots and found that it accelerates the increasing trend in the number of merged PRs, accelerates the decreasing trend in the number of closed PRs, decreases the number of comments in all PRs, increases the resolution time of merged PRs, and reverses the increasing trend in the resolution time of closed PRs. In another study, Wessel et al. [65] investigated the impact of adopting GitHub Actions and found that it decreases the number of both merged PRs and closed PRs, increases the number of discussion comments in merged PRs, decreases the number of discussion comments in closed PRs, increases the resolution time of merged PRs, and decreases the resolution time of closed PRs. However, our study focuses on investigating the impact of adopting Stale bot to better understand its helpfulness for pull-based development.

Review Latency and Decision of PRs. The influence of various technical, social, and personal factors on the review latency and decision of PRs has been extensively studied in the literature. Gousios et al. [19] was the first to investigate how technical factors can affect the merge decision and merge time of PRs. They found that the merge decision is mainly associated with whether the PR touches recently modified code, while the merge time is associated with the track record of the developer, as well as the size of the project, its test coverage, and its openness to external contributions. Gousios et al. [21] reports that the decision of integrators to accept a contribution is based on its quality, including conformance to the project style and architecture, source code quality, and test coverage. Tsay et al. [57] showed that in addition to technical factors, social factors could influence the acceptance of PRs. They found that while PRs with lots of comments are associated with a lower probability of getting accepted, the prior interactions of submitters in the project moderate this effect. Additionally, they found that well-established projects are more conservative while evaluating contributions.

Since then, many studies have continued investigating the role of different technical and social factors [34, 36, 45, 50, 69, 77], as well as various personal and demographic factors [9, 26, 41, 42, 48, 49, 56] on the review latency or decision of PRs. Recently, Zhang et al. [72] and Zhang et al. [71] conducted a large-scale empirical study of how a large set of factors identified through a systematic literature review can explain the review latency and decision of PRs, respectively. While these studies investigated what factors can make a PR get reviewed faster and accepted, our study investigates the characteristics of inactive PRs usually intervened by Stale bot to better understand its impacts.

3 DATASET

For our study, we need large and popular open-source projects as their higher workload makes them more likely to benefit significantly from the adoption of Stale bot. The projects should also have a rich history of using Stale bot as part of their pull-based development workflow to ensure that we have enough data for our analyses. To identify such projects, we rely on GH Archive [22],

Table 1. Overview of the projects selected to study the helpfulness of Stale bot for pull-based development.

Project	PRs	Stars	Contributors	Maintainers	Age in Months	Months Since Adoption	Domain	Language
nixos/nixpkgs	121,998	8,013	4,629	300	111	17	Package Manager	Nix
homebrew/homebrew-core	84,686	10,203	6,747	50	67	54	Package Manager	Ruby
ceph/ceph	43,885	9,826	1,573	73	120	35	Storage Platform	C++
automattic/wp-calypso	38,301	11,922	707	151	70	29	WordPress Frontend	JavaScript
home-assistant/core	34,922	47,437	3,875	51	96	19	Home Automation	Python
cleverraven/cataclysm-dda	33,239	5,762	1,686	36	107	26	Video Game	C++
homebrew/linuxbrew-core	23,259	1,171	285	13	64	47	Package Manager	Ruby
istio/istio	21,236	28,423	1,053	78	58	15	Microservice Mesh	Go
qgis/qgis	19,587	5,092	569	39	124	39	GIS System	C++
devexpress/devextreme	19,412	1,521	152	21	53	18	Web Framework	JavaScript
grafana/grafana	18,538	44,786	2,259	55	93	21	Visualization Platform	TypeScript
wikia/app	18,293	191	217	64	105	34	Wiki Engine	PHP
helm/charts	18,196	15,305	4,958	16	69	34	Kubernetes Apps	Go
grpc/grpc	17,939	32,361	1,221	49	81	25	RPC Framework	C++
solana-labs/solana	17,805	5,351	275	30	44	25	Decentralized Blockchain	Rust
home-assistant/home-assistant.io	17,780	2,253	4,410	28	81	46	Home Automation	HTML
conda-forge/staged-recipes	16,119	487	2,443	32	72	19	Package Manager	Python
apache/beam	15,924	5,068	961	33	68	38	Data Pipelines	Java
frappe/erpnext	15,840	9,970	580	42	122	39	ERP System	Python
riot-os/riot	14,335	3,985	447	33	104	26	Operating System	C

which archives all the public events happening on GitHub [11]. First, we query the GH Archive's public dataset on Google BigQuery [18] and look for all the events performed by Stale bot on PRs (i.e., the actor name is `stale[bot]` and the payload string contains `"pull_request"`). Then, we use the retrieved events to identify the projects that have ever used Stale bot in their pull-based development workflow. Next, we collect the timeline of activities for the identified projects using the `PyGithub` package [27] on November 17th, 2021. The timeline of activities of PRs is provided by the GitHub API [14] and includes the details (e.g., type, actor, and time) of all the events (e.g., commits, comments, labelings, and resolutions) that happened during their lifecycle [12, 13, 16].

We then determine the adoption time of each project by looking for the first event performed by Stale bot in the PRs of the project. Following the recommendation of Wagner et al. [58], we consider 12 months before and 12 months after the adoption (a total of two years) as our observation period. This period allows us to ensure that we have a sufficient number of observations for our analyses and to take into account the expected seasonal variations in the projects [58]. Therefore, we focus on the projects that have at least 12 months of pull-based development history before the adoption and that have also used Stale bot for at least 12 consecutive months in their pull-based development workflow after the adoption. Among these projects, we finally select the top 20 with the most PRs to focus on the largest and most popular projects.

Table 1 provides an overview of the projects that we selected for our study. In summary, the selected projects have thousands of PRs (median of 18,975), thousands of stars (median of 6,888), hundreds of contributors (median of 1,137), tens of maintainers (median of 41), years of pull-based development history (median of 81 months), and years of using Stale bot (median of 28 months). Additionally, these projects span multiple application domains and programming languages, providing a more diverse selection of projects for our study.

4 RQ_1: HOW MUCH DO THE STUDIED PROJECTS USE STALE BOT TO DEAL WITH THEIR PR BACKLOG?

Open-source projects on GitHub are adopting Stale bot to automatically follow up on the status of their inactive PRs, warn the contributors and maintainers about the lack of activity, and eventually close such PRs if there is no further progress on them [15, 29, 64]. As our first research question, we aim to understand the extent to which large open-source projects rely on Stale bot to automatically

deal with their unresolved PRs. In the following, we first explain our approach and then discuss our findings to answer this research question.

4.1 Approach

To investigate the usage of Stale bot, we analyze both its configuration and activity during its first year of adoption (i.e., our observation period) in the studied projects. For each project, we extract the configured number of days of inactivity before a PR is marked as stale (i.e., `daysUntilStale`) and the configured number of days of inactivity before a PR already marked as stale gets closed (i.e., `daysUntilClose`) from its configuration file of Stale bot (i.e., `.github/stale.yml`). We also measure the monthly activity of Stale bot in the PRs of each project as a proxy for the extent to which the project actually relies on Stale bot to deal with its unresolved PRs. Since Stale bot can either warn about the lack of activity in a PR or close a PR due to inactivity, we further distinguish between these two types of activities. For this purpose, we use the following definitions to classify Stale bot activities in a PR:

- **Intervention:** Any commenting, labeling, unlabeling, or closing event performed by Stale bot.
- **Warning:** Any commenting or labeling event performed by Stale bot that is not immediately followed by a closing event from Stale bot within a minute (to account for the processing delay between the closing comment and the actual closure of a PR on GitHub).
- **Closure:** A closing event performed by Stale bot.

4.2 Findings

Table 2 provides an overview of the usage of Stale bot during its first year of adoption across the studied projects, indicating (1) the number of open PRs immediately upon the adoption (Backlog), (2) the average number of open PRs at the beginning of each month (# Open PRs), (3) the average ratio of open PRs intervened by Stale bot in each month (% Intervened PRs), (4) the average ratio of open PRs warned by Stale bot in each month (% Warned PRs), (5) the average ratio of open PRs closed by Stale bot in each month (% Closed PRs), (6) the average configured number of days to stale a PR in each month (Days to Stale), and (7) the average configured number of days to close a stale PR in each month (Days to Close). Note that the ratio of intervened PRs might not equal the sum of the ratios of warned PRs and closed PRs in a project. This discrepancy arises because intervention includes other activities (i.e., unlabeling events) aside from warnings and closures, and also because PRs might get intervened multiple times within a month (e.g., closure after a warning).

For easier comparison, we group the projects based on the average monthly ratio of open PRs intervened by Stale bot (i.e., % Intervened PRs) into three levels of usage: (1) six projects have a high usage level where Stale bot intervened in more than 50% of monthly open PRs; (2) five projects have a moderate usage level where Stale bot intervened in between 25% and 50% of monthly open PRs; and (3) nine projects have a low usage level where Stale bot intervened in less than 25% of monthly open PRs. In the following, we discuss our findings in more detail.

The usage level of Stale bot widely varies among the projects. On average, Stale bot intervened between 2.9% and 70.8% of open PRs, warned between 2.5% and 65.5% of open PRs, and closed between 0% and 37.3% of open PRs each month in the projects. For example, in the nixos/nixpkgs project, which has both the largest PR backlog upon the adoption and the highest average monthly number of open PRs after the adoption among the studied projects, Stale bot only intervened in 6.9% of open PRs on average each month, and was not even configured to close any of them. In contrast, the solana-labs/solana project has the second lowest average monthly number of open PRs after the adoption among the studied projects, yet Stale bot warned 65.5% of open PRs and

Table 2. Usage of Stale bot during its first year of adoption across the studied projects. Dark gray highlights activities in more than 50% of monthly open PRs and light gray highlights activities in between 25% and 50% of monthly open PRs.

Usage Level	Project	Backlog	# Open PRs	Monthly Average % Intervened PRs	% Warned PRs	% Closed PRs	Days to Stale	Days to Close
High	solana-labs/solana	27	28	70.8%	65.5%	30.5%	15	7
	grafana/grafana	139	109	61.3%	48.4%	11.0%	14	30
	istio/istio	151	160	60.7%	47.3%	12.1%	14	27
	homebrew/linuxbrew-core	174	58	58.4%	51.8%	37.3%	21	7
	helm/charts	423	366	54.8%	43.8%	23.0%	30	14
	frappe/erpnext	14	44	51.0%	46.9%	13.7%	22	7
Moderate	qgis/qgis	40	31	48.0%	41.9%	18.4%	14	7
	wikia/app	23	24	46.1%	41.6%	19.8%	30	7
	homebrew/homebrew-core	72	82	32.5%	28.7%	9.0%	21	7
	conda-forge/staged-recipes	543	226	29.7%	16.7%	13.0%	150	30
	home-assistant/core	228	270	29.0%	26.8%	7.9%	65	7
Low	grpc/grpc	332	207	22.2%	17.3%	11.5%	150	6
	ceph/ceph	742	637	20.2%	15.0%	3.7%	60	90
	apache/beam	127	95	15.4%	13.2%	7.7%	60	7
	cleverraven/cataclysm-dda	65	95	13.2%	10.3%	1.6%	30	30
	riot-os/riot	555	454	11.0%	7.0%	3.7%	185	31
	devexpress/devextreme	31	31	10.6%	9.5%	3.9%	30	5
	home-assistant/home-assistant.io	53	65	7.6%	6.5%	3.7%	60	7
	nixos/nixpkgs	2,054	2,324	6.9%	5.3%	0.0%	180	–
	automattic/wp-calypso	293	349	2.9%	2.5%	1.8%	270	7

closed 30.5% of them on average each month. This wide range of activities indicates that while some projects rely conservatively on Stale bot, others rely on it aggressively to deal with their unresolved PRs.

The projects with a larger backlog of unresolved PRs have typically not relied more aggressively on Stale bot. The average monthly ratio of open PRs intervened by Stale bot is not significantly associated with the size of the PR backlog upon the adoption (Spearman's $\rho = -0.31$). Similarly, it is not significantly associated with the average monthly number of open PRs after the adoption (Spearman's $\rho = -0.38$). These results suggest that the higher activity of Stale bot in a project is not usually due to more accumulated unresolved PRs. Indeed, there is a significant negative correlation between the average monthly ratio of open PRs intervened by Stale bot and the average configured number of days to mark a PR as stale (Spearman's $\rho = -0.79$), suggesting that the higher activity of Stale bot in a project is usually due to a more aggressive configuration (i.e., fewer days of inactivity before a PR is marked as stale). In other words, the differences in the activity of Stale bot in different projects tend to be due to its configuration rather than the PR backlog size in a project.

Answer to RQ$_1$. We find that the usage level of Stale bot widely varies among the studied projects. On average each month, Stale bot intervened in less than 25% of open PRs in nine projects, between 25% and 50% of open PRs in five projects, and more than 50% of open PRs in six projects. Unexpectedly, the projects with a larger backlog of unresolved PRs have typically not relied more aggressively on Stale bot to deal with their unresolved PRs.

Table 3. Overview of the indicators measured to quantify the performance of the pull-based development workflow in the studied projects.

Dimension	Rationale	Indicator	Description
Resolved PRs	Stale bot closes inactive PRs and frees maintainers from manually tracking inactive PRs.	merged_pulls	Number of merged PRs within a month
		closed_pulls	Number of closed PRs within a month
Review Latency	Stale bot frees maintainers from triaging inactive PRs by automatically tracking, warning, and closing such PRs.	first_latency_m	Average first response latency (excluding Stale bot) of merged PRs within a month in hours
		first_latency_c	Average first response latency (excluding Stale bot) of closed PRs within a month in hours
		mean_latency_m	Average of the mean response latency (excluding Stale bot) of merged PRs within a month in hours
		mean_latency_c	Average of the mean response latency (excluding Stale bot) of closed PRs within a month in hours
Resolution Time	Stale bot prevents PRs from staying indefinitely open without any progress by closing inactive PRs.	resolution_time_m	Average resolution time of merged PRs within a month in hours
		resolution_time_c	Average resolution time of closed PRs within a month in hours
Review Discussion	Stale bot attracts the attention of participants by warning about the lack of activity.	comments_m	Average number of comments (excluding Stale bot) in merged PRs within a month
		comments_c	Average number of comments (excluding Stale bot) in closed PRs within a month
PR Updates	Stale bot prevents PRs from getting too outdated by warning about the lack of activity.	commits_m	Average number of commits in merged PRs within a month
		commits_c	Average number of commits in closed PRs within a month
Contributor Retention	Stale bot is frequently reported to frustrate the community by attempting to close PRs.	contributors	Number of active contributors within a month

5 RQ$_2$: WHAT IS THE IMPACT OF STALE BOT ON PULL-BASED DEVELOPMENT IN THE STUDIED PROJECTS?

In the previous research question, we found that the studied projects rely on Stale bot to deal with their accumulated backlog of unresolved PRs. As our second research question, we aim to understand if and how adopting Stale bot improves the efficiency and effectiveness of the pull-based development workflow in large open-source projects. In the following, we first explain our approach and then discuss our findings to answer this research question.

5.1 Approach

To investigate the impact of adopting Stale bot, we first need to quantify the performance of the pull-based development workflow in the studied projects. For this purpose, we consult similar studies that investigated the effect of an intervention on the pull-based development of open-source projects [62, 65]. As shown in Table 3, we measure 13 performance indicators covering six dimensions: (1) resolved PRs, (2) review latency, (3) resolution time, (4) review discussion, (5) PR updates, and (6) contributor retention.

Similar to previous studies [62, 65], we differentiate between the indicators of merged PRs (ending with _m) and closed PRs (ending with _c) to take into account the inherent differences in the characteristics of accepted and rejected PRs. To identify merged PRs, we resort to heuristics [31] similar to [28] because accepted PRs are not always merged using the standard methods provided through the GitHub interface. Accordingly, besides PRs with an explicit merged status (i.e., merged using the GitHub interface), we consider closed PRs with a commit inside the project that references them (e.g., "Close #123") as merged. For each month before and after the adoption, we then measure the indicators of each project by aggregating the data about PRs that have been merged or closed in that month's timeframe.

To estimate the potential impact of Stale bot on the indicators of the projects, we rely on interrupted time-series analysis (ITS) [58]. This method is a well-established quasi-experiment previously used in the software engineering literature to study the impact of an intervention (e.g., [62, 65, 73]). To estimate the longitudinal impact of an intervention (i.e., the adoption of Stale bot in our case), ITS compares a period before and after the intervention assuming the trend would be retained if the intervention had not occurred. To perform ITS in our study, we use the following linear regression model:

$$Y_t = \beta_0 + \beta_1 \times time_t + \beta_2 \times adoption_t + \beta_3 \times time\_since\_adoption_t + \beta_4 \times controls \quad (1)$$

where Y_t represents the value of indicator Y at time t; *time* represents the number of months passed since the start of the observation period at time t (encoded from 1 to 24 covering one year before

and one year after the adoption); *adoption* represents whether Stale bot has been adopted at time t (encoded as 0 before the adoption and as 1 after the adoption); *time_since_adoption* represents the number of months passed since the adoption of Stale bot at time t (encoded as 1 to 12 covering one year after the adoption and as 0 before the adoption); and *controls* includes a set of variables to control for the inherent differences among the studied projects. These control variables include: (1) the age of the project at the adoption time in months (denoted as *age_at_adoption*) as a proxy for the level of maturity in the project, (2) the number of PRs at the adoption time in the project (denoted as *pulls_at_adoption*) as a proxy for the level of activity in the project, (3) the number of contributors at the adoption time in the project (denoted as *contributors_at_adoption*) as a proxy for the size of the project community, and (4) the number of maintainers at the adoption time in the project (denoted as *maintainers_at_adoption*) as a proxy for the size of the project core team.

To implement the ITS regression in Equation (1), we build linear mixed-effects models [10, 67] using the `lmerTest` package [35]. To do so, we consider *age_at_adoption*, *pulls_at_adoption*, *contributors_at_adoption*, and *maintainers_at_adoption* as fixed effects and the name of the project (denoted as *project_name*) as the random intercept. While both these fixed-effect and random-effect variables aim to capture project-to-project variability, the random intercept allows for capturing unmeasured variability between the projects by assigning a different intercept to each project. We also log-transform all the variables with a skewed distribution to better satisfy the assumptions of linear mixed-effects models, such as linearity and homoscedasticity [67].

To evaluate the goodness of fit of our models, we measure the marginal and conditional coefficients of determination (R^2) [43] using the `performance` package [39]. While the marginal R^2 describes the proportion of the total variance explained only by the fixed effects, the conditional R^2 describes the proportion of the total variance explained by both the fixed and the random effects. To estimate the statistical significance of the coefficients in our models, we rely on the Satterthwaite's method [25] calculated using the `lmerTest` package [35]. We consider the traditional confidence level of 95% (i.e., $\alpha = 0.05$) [3] to identify significant variables. Additionally, to estimate the effect size of each variable, we measure its sum of squares based on type III analysis of variance (a.k.a. ANOVA) [17] using the `lmerTest` package [35]. The results describe the fraction of the total variance explained by the model that can be attributed to each variable.

To determine if the adoption of Stale bot has a potential impact on an indicator, we check the statistical significance of *time*, *adoption*, and *time_since_adoption* in the corresponding model. Specifically, a significant *time* shows the existing trend before the adoption, a significant *adoption* shows the change in level at the adoption time, and a significant *time_since_adoption* shows the change in the trend after the adoption. Accordingly, the sum of *time* and *time_since_adoption* represents the new trend after the adoption. To also estimate the effect size of the adoption when a significant change is observed, we rely on counterfactuals [58]: what would have happened if the adoption had never occurred and thus the trend before the adoption had continued unchanged. For this purpose, we assume that there has been neither a change in the level (i.e., *adoption* = 0) nor the trend (i.e., *time_since_adoption* = 0) and measure the change percentage compared to the actual predicted values of the model.

5.2 Findings

The results of the models to estimate the impact of Stale bot on different performance indicators are presented in Appendix A. For easier comparison, Figure 2 visualizes how the values of the impacted indicators vary each month during our observation period, along with the average of the model predictions (the solid red lines) and the average of the counterfactual predictions for all the studied projects (the dashed red lines). The variation plots for the remaining indicators can also be found in Appendix B. We observe that the projects closed more PRs within the first few months of

adopting Stale bot, but overall closed and merged fewer PRs afterward. The adoption of Stale bot is also associated with faster first reviews in merged PRs, faster resolutions in closed PRs, slightly fewer updates in merged PRs, and considerably fewer active contributors in the projects. In the following, we discuss our findings in more detail.

While more PRs are closed within the first few months of adopting Stale bot, overall fewer PRs are closed and merged afterward. As shown in Figure 2b, the number of closed PRs experienced an increase in the level but a decrease in the slope that reversed the increasing trend before the adoption. Specifically, our predictions indicate that 15% more PRs were closed in the first month of adoption but overall 10% fewer PRs were closed by the end of the first year of adoption. The short-term increase in the number of monthly closed PRs is expected as Stale bot closes accumulated inactive PRs immediately upon its adoption. From Figure 2a, we also observe a decrease in both the level and the slope of the number of merged PRs that decelerated the increasing trend before the adoption. While more PRs are still merged each month, our predictions indicate that overall 24% fewer PRs were merged by the end of the first year of adoption.

Merged PRs tend to have faster first reviews after the adoption of Stale bot. As shown in Figure 2c, the first review latency of merged PRs experienced a decrease in the slope that reversed the increasing trend before the adoption. Specifically, our predictions indicate that merged PRs had an overall 21% lower first review latency during the first year of adoption. However, the first review latency of closed PRs and the mean review latency of both closed PRs and merged PRs have not significantly changed after the adoption. Still, closed PRs tend to take longer to receive a first review over time, and this trend has not significantly changed after the adoption. The results suggest that the maintainers are focusing on PRs that are more likely to get merged, but at the cost of leaving the remaining PRs to linger until Stale bot closes them after a period of inactivity.

Closed PRs tend to have faster resolutions after the adoption of Stale bot. As shown in Figure 2d, the resolution time experienced a decrease in the slope across closed PRs that reversed the increasing trend before the adoption, but has not significantly changed across merged PRs. For example, our predictions indicate that closed PRs had 22% lower resolution time in the last month of the first year of adoption. The decrease in the resolution time of closed PRs is expected as the auto-close function of Stale bot does not allow PRs to stay open indefinitely without any activity.

The amount of discussions in PRs is not affected by the adoption of Stale bot. We find that the number of discussion comments has not significantly changed after the adoption of Stale bot across both merged PRs and closed PRs. This observation comes as a surprise since we expected Stale bot to encourage more communication during the review process of PRs by notifying the participants about the lack of activity and by reducing the workload of the maintainers.

Merged PRs tend to have slightly fewer updates after the adoption of Stale bot. As shown in Figure 2e, the total number of commits across merged PRs experienced a decrease in the slope but has not significantly changed across closed PRs. Specifically, our predictions indicate that merged PRs overall contained 11% fewer commits during the first year of adoption. Still, closed PRs tend to have more commits over time, and this trend has not significantly changed after the adoption.

The adoption of Stale bot is associated with a considerable decrease in the number of active contributors. As shown in Figure 2f, the number of monthly active contributors experienced a decrease in the slope that significantly decelerated the increasing trend before the adoption. Specifically, our predictions indicate that overall 14% fewer contributors were active each month during the first year of adoption. The decreased number of active contributors may reflect the frustration of the community regarding the usage of Stale bot in open-source projects.

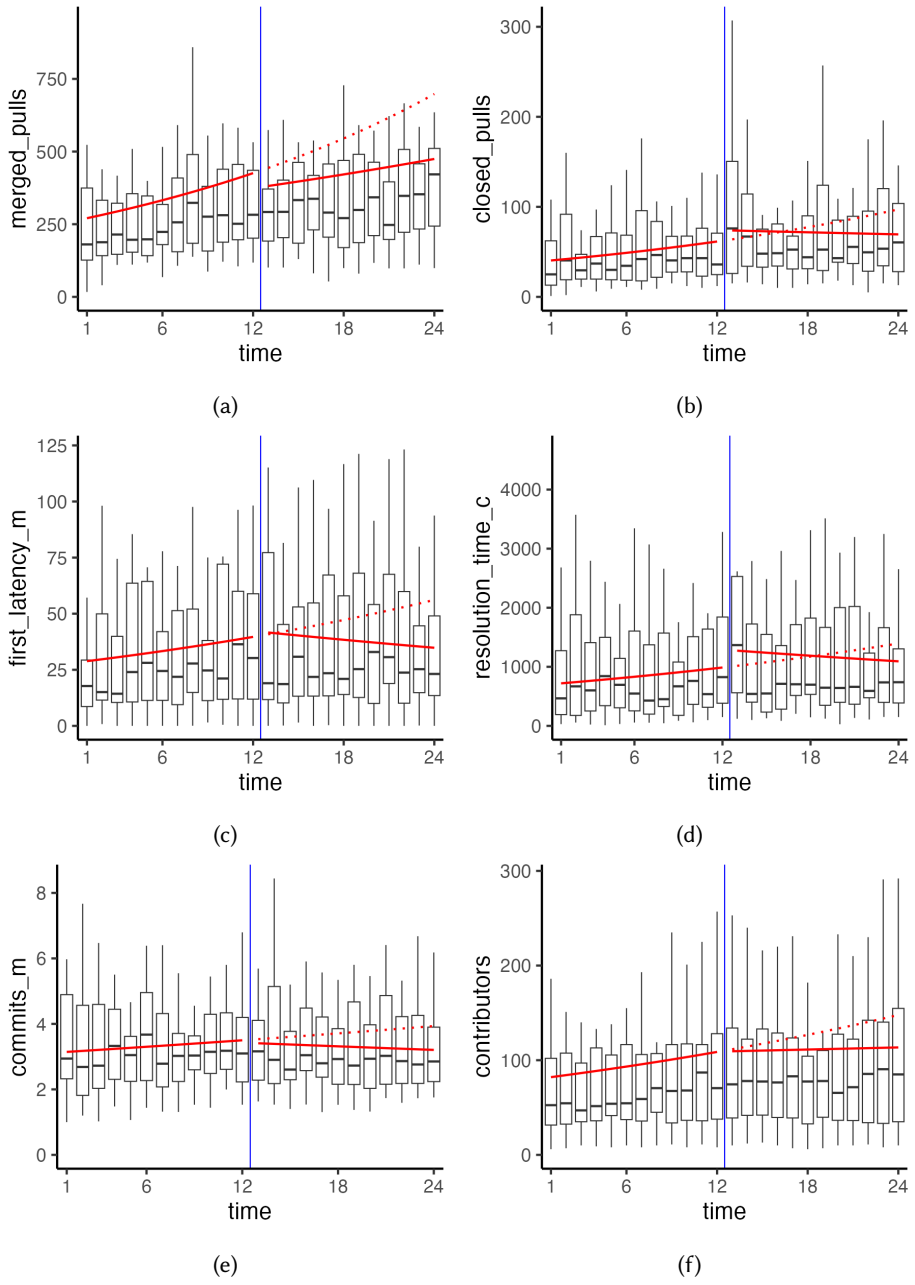


Fig. 2. Variation in (a) the number of merged PRs, (b) the number of closed PRs, (c) the first response latency of merged PRs, (d) the resolution time of closed PRs, (e) the number of commits in merged PRs, and (f) the number of active contributors each month during our observation period. The blue lines show the adoption time, the solid red lines show the average predictions, and the dashed red lines show the average counterfactual predictions of the model for all the studied projects. Note that we dropped the outliers in the plots and used the exponential function to convert the log-transformed output of the models back to real values.

Table 4. Overview of the factors measured to characterize PRs, their contributors, and their review processes.

Dimension	Factor	Description
Pull Request	description	Number of words in the title and description of the PR
	initial_commits	Number of commits at the submission time of the PR
	followup_commits	Number of commits after the submission time of the PR
	initial_changed_lines	Number of changed lines at the submission time of the PR
	followup_changed_lines	Number of changed lines after the submission time of the PR
	initial_changed_files	Number of changed files at the submission time of the PR
	followup_changed_files	Number of changed files after the submission time of the PR
Contributor	submitted_pulls	Number of previously submitted PRs by the contributor of the PR in the project
	acceptance_rate	Ratio of the previously merged PRs of the contributor of the PR in the project
	contribution_period	Number of months since the first submitted PR of the contributor of the PR in the project
Review Process	participants	Number of participants (excluding Stale bot) during the lifecycle of the PR
	participant_comments	Number of comments from the participants (excluding Stale bot) during the lifecycle of the PR
	contributor_comments	Number of comments from the contributor during the lifecycle of the PR
	first_latency	Number of hours till the first response of the participants (excluding Stale bot) in the PR
	mean_latency	Mean number of hours between the responses of the participants (excluding Stale bot) in the PR
	resolution_time	Number of hours between the submission and resolution times of the PR

Answer to RQ$_2$. We find that the studied projects closed more PRs within the first few months of adopting Stale bot, but overall closed and merged fewer PRs afterward. The adoption of Stale bot is also associated with faster first reviews in merged PRs, faster resolutions in closed PRs, slightly fewer updates in merged PRs, and considerably fewer active contributors in the projects.

6 RQ$_3$: WHAT KIND OF PRS ARE USUALLY INTERVENED BY STALE BOT IN THE STUDIED PROJECTS?

In the previous research question, we found that the adoption of Stale bot is associated with both positive and negative changes in the pull-based development workflow of the studied projects. As our last research question, we aim to understand what characteristics of PRs, their contributors, and their review processes are associated with a higher probability of getting intervened by Stale bot in large open-source projects. In the following, we first explain our approach and then discuss our findings to answer this research question.

6.1 Approach

To investigate the kinds of PRs intervened by Stale bot, we first need to characterize the PRs, their contributors, and their review processes. For this purpose, we consult the pull-based development literature [20, 70–72]. As shown in Table 4, we measure 16 factors covering three dimensions: (1) PR characteristics, (2) contributor characteristics, and (3) review process characteristics . After measuring these factors, we perform statistical analyses to determine how each factor differs in PRs that are intervened by Stale bot. To have a fair comparison between intervened and not intervened PRs, we filter our dataset to include only PRs that are closed or merged within the first year of adoption (i.e., our observation period). After this step, our dataset includes a total of 126,378 PRs, of which 6,833 PRs (~5.4%) have been intervened by Stale bot.

Then, we compare the distribution of the measured factors between intervened and not intervened PRs by generating violin plots [24] for each project using the `ggstatsplot` package [44]. The generated plots for each factor showing their median values (denoted by M), interquartile ranges (the box inside the violin), and probability densities (the width of the violin at each value) are presented in Appendix C. To test the statistical difference between the factors of intervened and

not intervened PRs, we apply the Mann–Whitney U test [40] as a nonparametric test that does not require the distribution of the factors to be normal. To perform this test, we use the `stats` package [46] and add the results to the generated plots. We consider the traditional confidence level of 95% (i.e., $\alpha = 0.05$) [3] to identify statistically significant factors.

While statistical significance verifies whether a difference exists between the factors of intervened and not intervened PRs, we also need to test their practical difference [32]. For this purpose, we use Cliff's delta [4] to estimate their magnitude of difference (i.e., effect size). The value of Cliff's delta (denoted by d) ranges from -1 to $+1$, where a positive d implies that the values of the factor in intervened PRs are usually greater than those of not intervened PRs, while a negative d implies the opposite. To calculate this statistic, we use the `effectsize` package [1] and add the results to the generated plots. Finally, for easier comparison, we convert the d values to qualitative magnitudes according to the following thresholds as suggested by Hess and Kromrey [23]:

$$\text{Effect size} = \begin{cases} \text{Negligible}, & \text{if } |d| \leq 0.147 \\ \text{Small}, & \text{if } 0.147 < |d| \leq 0.33 \\ \text{Medium}, & \text{if } 0.33 < |d| \leq 0.474 \\ \text{Large}, & \text{if } 0.474 < |d| \leq 1 \end{cases}$$

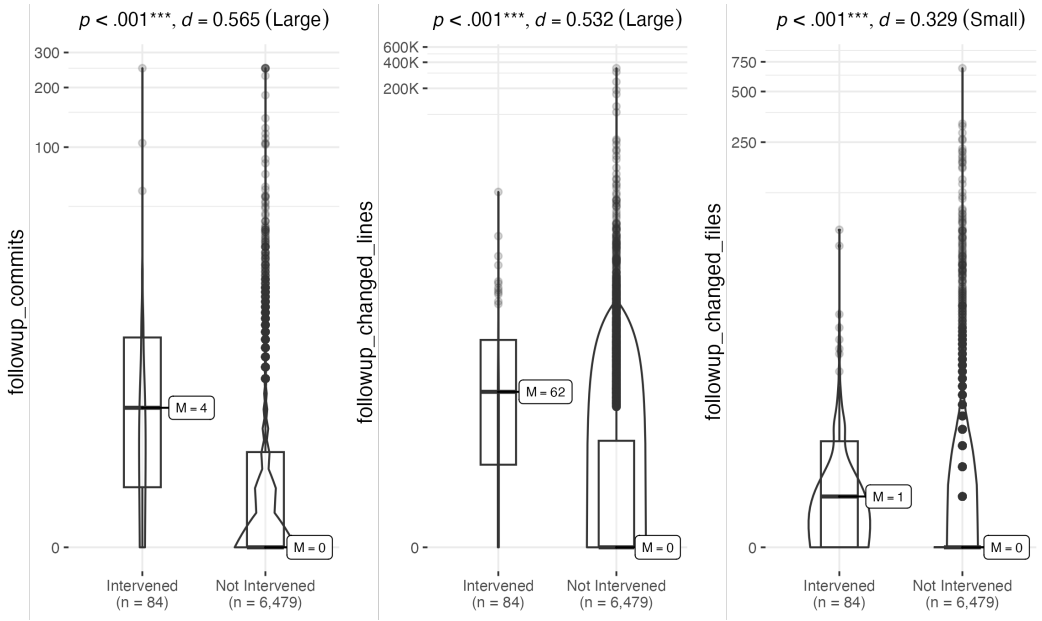
6.2 Findings

Table 5 summarizes the significant differences in the characteristics of PRs with and without intervention from Stale bot across the studied projects. For each factor, the table shows the number of projects in which intervened PRs tend to have significantly higher values compared to not intervened PRs, the number of projects in which intervened PRs and not intervened PRs are not significantly different, and the number of projects in which intervened PRs tend to have significantly lower values compared to not intervened PRs. We consider a factor significant if its difference between intervened and not intervened PRs is both statistically significant (i.e., $p < 0.05$) and practically significant (i.e., the effect size is small, medium, or large). We observe the most common differences in the characteristics of contributors and the review processes of PRs. Specifically, the largest differences are in the resolution time of PRs (higher in all the projects), the number of prior PRs by contributors (lower in all the projects), the mean response latency of PRs (higher in 19 projects), the acceptance rate of contributors (lower in 19 projects), and the contribution period of contributors (lower in 18 projects). In the following, we discuss our findings in more detail.

PRs intervened by Stale bot tend to be more complex. As shown in the PR dimension of Table 5, intervened PRs tend to significantly include fewer commits (8 projects), contain smaller changes (3 projects), and touch fewer files (7 projects) upon submission compared to not intervened PRs. However, after the submission, intervened PRs tend to significantly include more commits (14 projects), contain larger changes (14 projects), and touch more files (11 projects). For example, the intervened PRs in the cleverraven/cataclysm-dda project on median have 4 more commits, 62 more changed lines, and 1 more changed file after their submission (see Figure 3). Regarding the description length, we also observe that 6 projects tend to have lengthier descriptions and 4 projects tend to have shorter descriptions in intervened PRs. While the description length of PRs rarely changes during the review process of a PR, the size of changes (i.e., the number of commits, changed lines, or changed files) is likely to increase as the PR gets iteratively reviewed and updated. The results indicate that PRs mostly intervened by Stale bot have received considerable effort from their contributors to make it ready to get merged into the project.

Table 5. Differences in the characteristics of PRs with and without intervention from Stale bot across the studied projects.

Dimension	Factor	Higher in Intervened PRs			No Difference	Lower in Intervened PRs		
		Small	Medium	Large		Small	Medium	Large
Pull Request	description	4	1	1	10	3	–	1
	initial_commits	–	–	–	12	7	–	1
	followup_commits	8	4	2	6	–	–	–
	initial_changed_lines	1	–	–	16	2	–	1
	followup_changed_lines	8	4	2	6	–	–	–
	initial_changed_files	–	–	–	13	6	–	1
	followup_changed_files	9	2	–	9	–	–	–
Contributor	submitted_pulls	–	–	–	–	3	14	3
	acceptance_rate	–	–	–	1	6	11	2
	contribution_period	–	–	–	2	12	5	1
Review Process	participants	5	–	8	4	2	1	–
	participant_comments	4	6	4	3	–	2	1
	contributor_comments	5	6	3	6	–	–	–
	first_latency	2	5	6	4	2	–	1
	mean_latency	–	2	17	1	–	–	–
	resolution_time	–	–	20	–	–	–	–



(a) (b) (c)

Fig. 3. Differences between PRs with and without intervention from Stale bot in the cleverraven/cataclysm-dda project regarding (a) the number of follow-up commits, (b) the number of follow-up changed lines, and (c) the number of follow-up changed files after the submission.

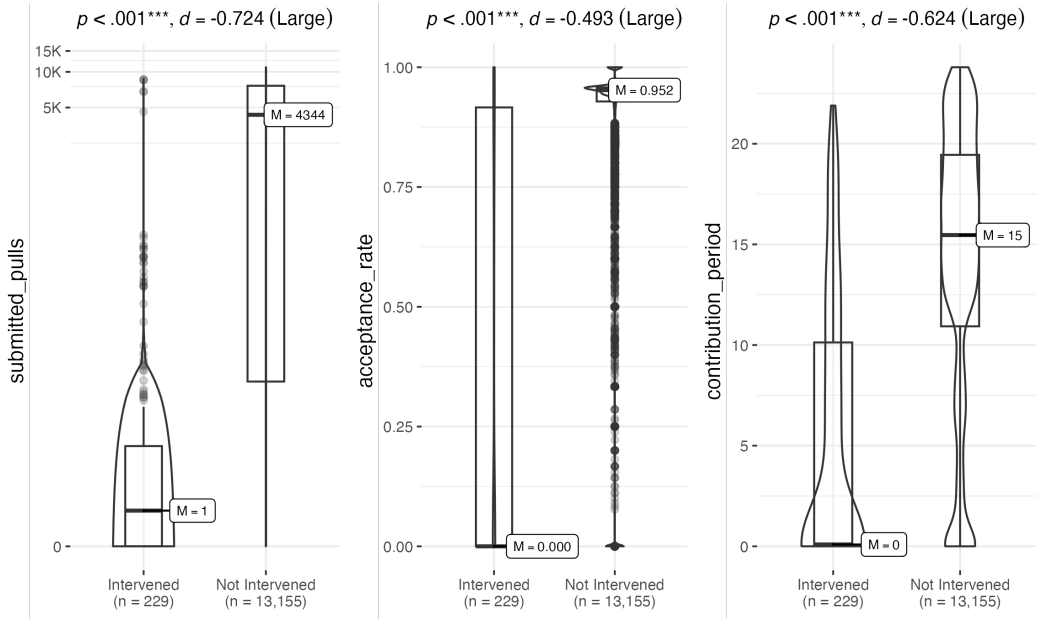


Fig. 4. Differences between the contributors of PRs with and without intervention from Stale bot in the homebrew/homebrew-core project regarding (a) the number of prior PRs, (b) the acceptance rate, and (c) the contribution period.

The contributors of PRs intervened by Stale bot tend to be less experienced. As shown in the contributor dimension of Table 5, the contributors of intervened PRs tend to significantly have previously submitted fewer PRs (all the projects), have a lower acceptance rate (19 projects), and have a lower contribution period (18 projects) compared to the contributors of not intervened PRs. For example, the contributors of intervened PRs in the homebrew/homebrew-core project on median have previously submitted 4,343 fewer PRs, have a 0.95 lower rate of acceptance, and have contributed to the project for 15 fewer months (see Figure 4). However, novice contributors are the ones who face the most barriers and thus need the most guidance from the maintainers [51–53]. The lack of responsiveness from the reviewers is also cited as a major reason why contributors (especially novice or casual contributors) leave the review processes of PRs unfinished and even stop further contributing to the project [30, 37, 53, 59]. Therefore, the results indicate that while Stale bot can help projects automatically deal with their inactive PRs, it may also lead to lower engagement of the community and eventually contributor abandonment, especially if the reviewers' input is required to continue the review process.

The review processes of PRs intervened by Stale bot tend to be lengthier. As shown in the review process dimension of Table 5, the review processes of intervened PRs tend to significantly involve more participants (13 projects), receive more comments from the participants (14 projects), receive more comments from the contributors (14 projects), take longer to receive their first review (13 projects), take longer to receive a follow-up review (19 projects), and take longer to get resolved (all the projects) compared to the review processes of not intervened PRs. For example, the review processes of intervened PRs in the homebrew/homebrew-core project on median involve 2 more

participants, receive 5 more comments from the participants, receive 2 more comments from the contributors, takes 14 more hours to receive their first review, takes 146 more hours to receive a follow-up review, and takes 1,168 more hours to get resolved (see Figure 5). While a longer resolution time in intervened PRs is expected as Stale bot intervenes in PRs that have been idle for a while, a longer time to receive reviews (both first and follow-up) from the participants is rather interesting.

Answer to RQ_3. We find that Stale bot tends to intervene more in complex PRs, PRs from novice contributors, and PRs with lengthy review processes. Specifically, besides the resolution time of PRs, the largest differences are observed in the number of prior PRs by contributors, the mean response latency of PRs, the acceptance rate of contributors, and the contribution period of contributors.

7 IMPLICATIONS

In the following, we combine our findings to further discuss the implications of our study.

Stale bot can help projects deal with a backlog of unresolved PRs. Almost all the studied projects relied on Stale bot to automatically close their accumulated PRs after a period of inactivity. In fact, the projects closed more PRs within the first few months of adopting Stale bot, yet closed and even merged considerably fewer PRs afterward. These findings suggest that adopting Stale bot could be an effective strategy for quickly dealing with a backlog of unresolved PRs in the short term. However, projects should be aware that the rule-based nature of Stale bot could wrongly close PRs that may still be under progress despite being inactive for some time [63]. Moreover, the automatic closure of PRs by Stale bot is known to raise the most negative reactions from the contributors and participants, especially when they perceive the closure as erroneous or unjustified [7].

Stale bot can help projects improve the review process of PRs. After the adoption of Stale bot, PRs that ended up being merged received faster reviews after their submission, and PRs that ended up being closed were also resolved much faster in the studied projects. However, PRs that end up being closed are increasingly taking longer to receive a review from the reviewers. The adoption of Stale bot did not improve this trend or encourage more communication during the review process of PRs. These findings suggest that maintainers are focusing on PRs that are more likely to get merged, but at the cost of leaving the remaining PRs to linger until Stale bot closes them after a period of inactivity. However, projects should be mindful that such a strategy will likely result in fewer merged PRs over time.

Stale bot can negatively affect the contributors of projects. The studied projects experienced a considerable decrease in their number of active contributors after the adoption of Stale bot. While Stale bot can help projects automatically deal with their inactive PRs, it is also more likely to intervene in PRs submitted by less experienced contributors. Projects should ensure that PRs receive timely reviews and responses from their reviewers, particularly if their input is needed to continue the review process. This is particularly important as the lack of responsiveness from reviewers is cited as a major reason why contributors (especially novice or casual contributors) leave the review process of PRs unfinished and even stop further contributing to a project [30, 37, 53, 59]. Therefore, implicitly ignoring unresolved PRs till Stale bot eventually closes them may lead to decreased community engagement and an increased probability of contributor abandonment.

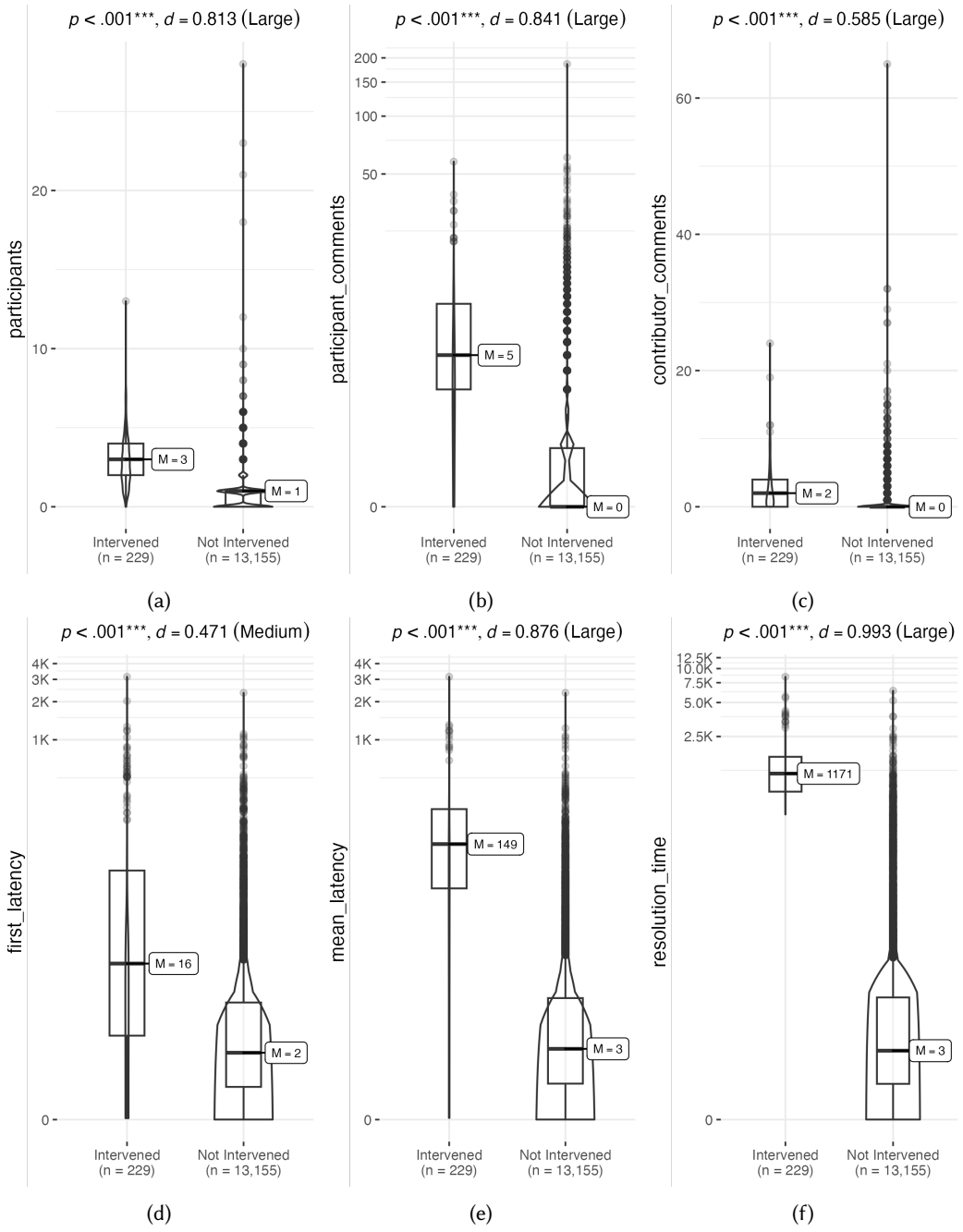


Fig. 5. Differences between the review processes of PRs with and without intervention from Stale bot in the homebrew/homebrew-core project regarding (a) the number of participants, (b) the number of comments from the participants, (c) the number of comments from the contributors, (d) the first review latency, (e) the mean review latency, and (f) the resolution time.

8 LIMITATIONS

In the following, we discuss threats to the internal, construct, and external validity [33] of our study and explain the measures taken to mitigate them.

Construct Validity. Construct validity is concerned with how accurately we quantified the helpfulness of Stale bot for pull-based development. We measured 13 different performance indicators covering various dimensions, including resolved PRs, review latency, resolution time, review discussion, PR updates, and contributor retention. To identify these indicators, we consulted similar studies that investigated the effects of interventions on the pull-based development of projects [62, 65] and also drew from our previous experience studying abandoned PRs [30]. Nevertheless, projects may experience changes in aspects that we did not consider or that are difficult to quantify, such as code quality. Forsgren et al. [8] describes various facets of productivity for individual developers and software teams, which future studies can consult for a more comprehensive investigation of the helpfulness of Stale bot.

Internal Validity. Internal validity is concerned with whether the adoption of Stale bot has actually caused the observed effects. Although we applied interrupted time-series analysis as a well-established quasi-experiment to understand the impact of adopting Stale bot, we cannot conclusively claim that Stale bot caused the observed changes. It is possible that the adoption of Stale bot in a project occurred simultaneously with other events that could explain the observed changes, such as the introduction of new tools or practices. For example, less than a month before adopting Stale bot, the devexpress/devextreme project modified its continuous integration (CI) workflow to also run on submitted PRs in addition to pushed commits [76]. To mitigate this threat, we utilized mixed-effects models for implementing the interrupted time-series analysis. This approach allows us to identify changes that are commonly observed across the majority of projects rather than changes that are specific to only a few projects. In other words, it is unlikely that the majority of the studied projects experienced significant changes to their pull-based development workflow during the same month they adopted Stale bot.

External Validity. External validity is concerned with how well our findings may be generalized to other open-source projects. In this study, we aimed to evaluate the helpfulness of Stale bot in open-source projects, specifically concerning their pull-based development. To conduct our study, we selected 20 large and popular open-source projects with a rich history of using Stale bot in their pull-based development workflow. While we believe these projects are more likely to benefit from adopting Stale bot, we recognize that they cannot represent the entire open-source ecosystem. In other words, the selected projects may not be representative of other open-source projects with different characteristics, such as their size, maturity, popularity, workload, culture, and development practices. Therefore, the findings of this study may not apply to all open-source projects. To acquire more broadly applicable insights, future research can replicate this study using a more diverse selection of projects.

9 CONCLUSION

PRs that are neither progressed nor resolved accumulate over time, clutter the list of PRs, and eventually make it difficult for the maintainers to manage and prioritize unresolved PRs. To automatically track such inactive PRs, follow up on their progress, and close them if needed, Stale bot was introduced by GitHub. Despite its increasing adoption, there are ongoing debates on whether using Stale bot alleviates or exacerbates the problem of inactive PRs. To better understand if and how Stale bot helps open-source projects in their pull-based development workflow, we conducted an empirical study of 20 large and popular open-source projects. First, we analyzed

the configuration and activity of Stale bot to understand the extent to which the projects relied on Stale bot to automatically deal with their unresolved PRs. Then, we applied interrupted time-series analysis as a well-established quasi-experiment to understand if and how adopting Stale bot improved the efficiency and effectiveness of the pull-based development workflow in the projects. Next, we analyzed the characteristics of PRs, their contributors, and their review processes to understand the factors that were associated with a higher probability of getting intervened by Stale bot in the projects. Finally, we combined our observations to discuss the potential benefits and drawbacks of employing Stale bot within a pull-based development workflow. In summary, we found that Stale bot can help projects deal with a backlog of unresolved PRs and also improve the review process of PRs. However, the adoption of Stale bot can negatively affect the contributors (especially novice or casual contributors) in a project.

REFERENCES

[1] Mattan S. Ben-Shachar, Daniel Lüdecke, and Dominique Makowski. 2020. effectsize: estimation of effect size indices and standardized parameters. *Journal of Open Source Software* 5, 56 (2020), 1–7. https://doi.org/10.21105/joss.02815

[2] Nathan Cassee, Bogdan Vasilescu, and Alexander Serebrenik. 2020. The silent helper: the impact of continuous integration on code reviews. In *Proceedings of the 27th International Conference on Software Analysis, Evolution, and Reengineering (SANER 2020)*. 423–434. https://doi.org/10.1109/SANER48275.2020.9054818

[3] David Chavalarias, Joshua David Wallach, Alvin Ho Ting Li, and John P. A. Ioannidis. 2016. Evolution of reporting *P* values in the biomedical literature, 1990-2015. *JAMA* 315, 11 (2016), 1141–1148. https://doi.org/10.1001/jama.2016.1952

[4] Norman Cliff. 1993. Dominance statistics: ordinal analyses to answer ordinal questions. *Psychological Bulletin* 114, 3 (1993), 494–509. https://doi.org/10.1037/0033-2909.114.3.494

[5] John W. Creswell and J. David Creswell. 2017. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (5th ed.). SAGE Publications, Inc. https://us.sagepub.com/en-us/nam/research-design/book255675

[6] Drew DeVault. 2021. GitHub Stale bot considered harmful. https://drewdevault.com/2021/10/26/stalebot.html

[7] Juan Carlos Farah, Basile Spaenlehauer, Xinyang Lu, Sandy Ingram, and Denis Gillet. 2022. An exploratory study of reactions to bot comments on GitHub. In *Proceedings of the 4th International Workshop on Bots in Software Engineering (BotSE 2022)*. 18–22. https://doi.org/10.1145/3528228.3528409

[8] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of developer productivity: there's more to it than you think. *Queue* 19, 1 (2021), 20–48. https://doi.org/10.1145/3454122.3454124

[9] Leonardo B. Furtado, Bruno Cartaxo, Christoph Treude, and Gustavo Pinto. 2021. How successful are open source contributions from countries with different levels of human development? *IEEE Software* 38, 2 (2021), 58–63. https://doi.org/10.1109/MS.2020.3044020

[10] Andrzej Gałecki and Tomasz Burzykowski. 2013. *Linear Mixed-Effects Models Using R: A Step-by-Step Approach*. Springer. https://doi.org/10.1007/978-1-4614-3900-4

[11] GitHub. 2023. Events - GitHub Docs. https://docs.github.com/en/rest/activity/events

[12] GitHub. 2023. Issues - GitHub Docs. https://docs.github.com/en/rest/reference/issues

[13] GitHub. 2023. Pulls - GitHub Docs. https://docs.github.com/en/rest/reference/pulls

[14] GitHub. 2023. REST API - GitHub Docs. https://docs.github.com/en/rest

[15] GitHub. 2023. Stale - GitHub Marketplace. https://github.com/marketplace/stale

[16] GitHub. 2023. Timeline - GitHub Docs. https://docs.github.com/en/rest/issues/timeline

[17] James Howard Goodnight. 1980. Tests of hypotheses in fixed effects linear models. *Communications in Statistics - Theory and Methods* 9, 2 (1980), 167–180. https://doi.org/10.1080/03610928008827869

[18] Google. 2023. BigQuery: cloud data warehouse. https://cloud.google.com/bigquery

[19] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. 345–355. https://doi.org/10.1145/2568225.2568260

[20] Georgios Gousios and Andy Zaidman. 2014. A dataset for pull-based development research. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. 368–371. https://doi.org/10.1145/2597073.2597122

[21] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. 2015. Work practices and challenges in pull-based development: the integrator's perspective. In *Proceedings of the IEEE/ACM 37th International Conference on Software Engineering (ICSE 2015)*. 358–368. https://doi.org/10.1109/ICSE.2015.55

[22] Ilya Grigorik. 2023. GH Archive: A project to record the public GitHub timeline, archive it, and make it easily accessible for further analysis. https://www.gharchive.org

[23] Melinda R. Hess and Jeffrey D. Kromrey. 2004. Robust confidence intervals for effect sizes: a comparative study of Cohen's d and Cliff's delta under non-normality and heterogeneous variances. In *Presented at the Annual Meeting of the American Educational Research Association (AERA 2004)*. 1–30. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.487.8299

[24] Jerry L. Hintze and Ray D. Nelson. 1998. Violin plots: a box plot-density trace synergism. *The American Statistician* 52, 2 (1998), 181–184. https://doi.org/10.1080/00031305.1998.10480559

[25] Alex Hrong-Tai Fai and Paul L. Cornelius. 1996. Approximate F-tests of multiple degree of freedom hypotheses in generalized least squares analyses of unbalanced split-plot experiments. *Journal of Statistical Computation and Simulation* 54, 4 (1996), 363–378. https://doi.org/10.1080/00949659608811740

[26] Rahul N. Iyer, S. Alex Yun, Meiyappan Nagappan, and Jesse Hoey. 2021. Effects of personality traits on pull request acceptance. *IEEE Transactions on Software Engineering* 47, 11 (2021), 2632–2643. https://doi.org/10.1109/TSE.2019.2960357

[27] Vincent Jacques. 2023. PyGithub: typed interactions with the GitHub API v3. https://github.com/PyGithub/PyGithub

[28] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21, 5 (2016), 2035–2071. https://doi.org/10.1007/s10664-015-9393-5

[29] Brandon Keepers. 2023. Stale - GitHub Repository. https://github.com/probot/stale

[30] SayedHassan Khatoonabadi, Diego Elias Costa, Rabe Abdalkareem, and Emad Shihab. 2023. On wasted contributions: understanding the dynamics of contributor-abandoned pull requests – a mixed-methods study of 10 large open-source projects. *ACM Transactions on Software Engineering and Methodology* 32, 1 (2023), 1–39. https://doi.org/10.1145/3530785

[31] SayedHassan Khatoonabadi, Shahriar Lotfi, and Ayaz Isazadeh. 2021. GAP2WSS: a genetic algorithm based on the Pareto principle for web service selection. https://doi.org/10.48550/arXiv.2109.10430

[32] Roger E. Kirk. 1996. Practical significance: a concept whose time has come. *Educational and Psychological Measurement* 56, 5 (1996), 746–759. https://doi.org/10.1177/0013164496056005002

[33] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-Based Software Engineering and Systematic Reviews* (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/b19467

[34] Oleksii Kononenko, Tresa Rose, Olga Baysal, Michael Godfrey, Dennis Theisen, and Bart de Water. 2018. Studying pull request merges: a case study of Shopify's Active Merchant. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP 2018)*. 124–133. https://doi.org/10.1145/3183519.3183542

[35] Alexandra Kuznetsova, Per B. Brockhoff, and Rune H. B. Christensen. 2017. lmerTest package: tests in linear mixed effects models. *Journal of Statistical Software* 82, 13 (2017), 1–26. https://doi.org/10.18637/jss.v082.i13

[36] Valentina Lenarduzzi, Vili Nikkola, Nyyti Saarimäki, and Davide Taibi. 2021. Does code quality affect pull request acceptance? An empirical study. *Journal of Systems and Software* 171 (2021), 1–14. https://doi.org/10.1016/j.jss.2020.110806

[37] Zhixing Li, Yue Yu, Tao Wang, Gang Yin, ShanShan Li, and Huaimin Wang. 2022. Are you still working on this? An empirical study on pull request abandonment. *IEEE Transactions on Software Engineering* 48, 6 (2022), 2173–2188. https://doi.org/10.1109/TSE.2021.3053403

[38] Dongyu Liu, Micah J. Smith, and Kalyan Veeramachaneni. 2020. Understanding user-bot interactions for small-scale automation in open-source development. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA 2020)*. 1–8. https://doi.org/10.1145/3334480.3382998

[39] Daniel Lüdecke, Mattan S. Ben-Shachar, Indrajeet Patil, Philip Waggoner, and Dominique Makowski. 2021. performance: an R package for assessment, comparison and testing of statistical models. *Journal of Open Source Software* 6, 60 (2021), 1–8. https://doi.org/10.21105/joss.03139

[40] Henry B. Mann and Donald R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18, 1 (1947), 50–60. https://doi.org/10.1214/aoms/1177730491

[41] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. 2021. Insights into nonmerged pull requests in GitHub: is there evidence of bias based on perceptible race? *IEEE Software* 38, 2 (2021), 51–57. https://doi.org/10.1109/MS.2020.3036758

[42] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. 2022. On the relationship between the developer's perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering* 48, 8 (2022), 2955–2968. https://doi.org/10.1109/TSE.2021.3073773

[43] Shinichi Nakagawa, Paul C. D. Johnson, and Holger Schielzeth. 2017. The coefficient of determination R^2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. *Journal of The Royal Society Interface* 14, 134 (2017), 1–11. https://doi.org/10.1098/rsif.2017.0213

[44] Indrajeet Patil. 2021. Visualizations with statistical details: the 'ggstatsplot' approach. *Journal of Open Source Software* 6, 61 (2021), 1–5. https://doi.org/10.21105/joss.03167

[45] Gustavo Pinto, Luiz Felipe Dias, and Igor Steinmacher. 2018. Who gets a patch accepted first? Comparing the contributions of employees and volunteers. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2018)*. 110–113. https://doi.org/10.1145/3195836.3195858

[46] R Core Team. 2023. R: a language and environment for statistical computing. https://www.R-project.org

[47] Akond Rahman, Farzana Ahamed Bhuiyan, Mohammad Mehedi Hassan, Hossain Shahriar, and Fan Wu. 2022. Towards automation for MLOps: an exploratory study of bot usage in deep learning libraries. In *Proceedings of the IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC 2022)*. 1093–1097. https://doi.org/10.1109/COMPSAC54236.2022.00171

[48] Ayushi Rastogi. 2016. Do biases related to geographical location influence work-related decisions in GitHub?. In *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE-C 2016)*. 665–667. https://doi.org/10.1145/2889160.2891035

[49] Ayushi Rastogi, Nachiappan Nagappan, Georgios Gousios, and André van der Hoek. 2018. Relationship between geographical location and evaluation of developer contributions in GitHub. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2018)*. 1–8. https://doi.org/10.1145/3239235.3240504

[50] Daricélio Moreira Soares, Manoel Limeira de Lima Júnior, Leonardo Murta, and Alexandre Plastino. 2015. Acceptance factors of pull requests in open-source projects. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC 2015)*. 1541–1546. https://doi.org/10.1145/2695664.2695856

[51] Igor Steinmacher, Ana Paula Chaves, Tayana Uchoa Conte, and Marco Aurélio Gerosa. 2014. Preliminary empirical identification of barriers faced by newcomers to open source software projects. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES 2014)*. 51–60. https://doi.org/10.1109/SBES.2014.9

[52] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW 2015)*. 1379–1392. https://doi.org/10.1145/2675133.2675215

[53] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Why do newcomers abandon open source software projects?. In *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013)*. 25–32. https://doi.org/10.1109/CHASE.2013.6614728

[54] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting developer productivity one bot at a time. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. 928–931. https://doi.org/10.1145/2950290.2983989

[55] Poul Kjeldager Sørensen. 2014. Pull Request #2143 - DefinitelyTyped/DefinitelyTyped. https://github.com/DefinitelyTyped/DefinitelyTyped/pull/2143

[56] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. 2017. Gender differences and bias in open source: pull request acceptance of women versus men. *PeerJ Computer Science* 2017, 3 (2017), 1–30. https://doi.org/10.7717/peerj-cs.111

[57] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. 356–366. https://doi.org/10.1145/2568225.2568315 Issue: 1.

[58] Anita K. Wagner, Stephen B. Soumerai, Fang Zhang, and Dennis Ross-Degnan. 2002. Segmented regression analysis of interrupted time series studies in medication use research. *Journal of Clinical Pharmacy and Therapeutics* 27, 4 (2002), 299–309. https://doi.org/10.1046/j.1365-2710.2002.00430.x

[59] Qingye Wang, Xin Xia, David Lo, and Shanping Li. 2019. Why is my code change abandoned? *Information and Software Technology* 110 (2019), 108–120. https://doi.org/10.1016/j.infsof.2019.02.007

[60] Mairieli Wessel, Ahmad Abdellatif, Igor Wiese, Tayana Conte, Emad Shihab, Marco Aurélio Gerosa, and Igor Steinmacher. 2022. Bots for pull requests: the good, the bad, and the promising. In *Proceedings of the 44th International Conference on Software Engineering (ICSE 2022)*. 274–286. https://doi.org/10.1145/3510003.3512765

[61] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco Aurélio Gerosa. 2018. The power of bots: characterizing and understanding bots in OSS projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–19. https://doi.org/10.1145/3274451

[62] Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco Aurélio Gerosa. 2022. Quality gatekeepers: investigating the effects of code review bots on pull request activities. *Empirical Software Engineering* 27, 5 (2022), 1–36. https://doi.org/10.1007/s10664-022-10130-9

[63] Mairieli Wessel and Igor Steinmacher. 2020. The inconvenient side of software bots on pull requests. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW 2020)*. 51–55. https://doi.org/10.1145/3387940.3391504

[64] Mairieli Wessel, Igor Steinmacher, Igor Wiese, and Marco A. Gerosa. 2019. Should I stale or should I close? An analysis of a bot that closes abandoned issues and pull requests. In *Proceedings of the IEEE/ACM 1st International Workshop on*

Bots in Software Engineering (BotSE 2019). 38–42. https://doi.org/10.1109/BotSE.2019.00018

[65] Mairieli Wessel, Joseph Vargovich, Marco Aurélio Gerosa, and Christoph Treude. 2022. GitHub Actions: the impact on the pull request process. https://doi.org/10.48550/arXiv.2206.14118

[66] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco Aurélio Gerosa. 2021. Don't disturb me: challenges of interacting with software bots on open source software projects. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–21. https://doi.org/10.1145/3476042

[67] Brady T. West, Kathleen B. Welch, and Andrzej T. Gałecki. 2014. *Linear Mixed Models: A Practical Guide Using Statistical Software* (2nd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/b17198

[68] Ben Winding. 2021. GitHub Stale bots: a false economy. https://blog.benwinding.com/github-stale-bots/index.html

[69] Yue Yu, Gang Yin, Tao Wang, Cheng Yang, and Huaimin Wang. 2016. Determinants of pull-based development in the context of continuous integration. *Science China Information Sciences* 59, 8 (2016), 1–14. https://doi.org/10.1007/s11432-016-5595-8

[70] Xunhui Zhang, Ayushi Rastogi, and Yue Yu. 2020. On the shoulders of giants: a new dataset for pull-based development research. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR 2020)*. 543–547. https://doi.org/10.1145/3379597.3387489

[71] Xunhui Zhang, Yue Yu, Georgios Gousios, and Ayushi Rastogi. 2023. Pull request decisions explained: an empirical overview. *IEEE Transactions on Software Engineering* 49, 2 (2023), 849–871. https://doi.org/10.1109/TSE.2022.3165056

[72] Xunhui Zhang, Yue Yu, Tao Wang, Ayushi Rastogi, and Huaimin Wang. 2022. Pull request latency explained: an empirical overview. *Empirical Software Engineering* 27, 6 (2022), 1–38. https://doi.org/10.1007/s10664-022-10143-4

[73] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. 2017. The impact of continuous integration on other software development practices: a large-scale empirical study. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*. 60–71. https://doi.org/10.1109/ASE.2017.8115619

[74] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: willingness and opportunity in OSS community. In *Proceedings of the 34th International Conference on Software Engineering (ICSE 2012)*. 518–528. https://doi.org/10.1109/ICSE.2012.6227164

[75] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. 2016. Effectiveness of code contribution: from patch-based to pull-request-based tools. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. 871–882. https://doi.org/10.1145/2950290.2950364

[76] Alexander Ziborov. 2020. Commit #439b8735fa93709ec32602bb32944bf9214ce785 - DevExpress/DevExtreme. https://github.com/DevExpress/DevExtreme/commit/439b8735fa93709ec32602bb32944bf9214ce785

[77] Weiqin Zou, Jifeng Xuan, Xiaoyuan Xie, Zhenyu Chen, and Baowen Xu. 2019. How does code style inconsistency affect pull request integration? An exploratory study on 117 GitHub projects. *Empirical Software Engineering* 24, 6 (2019), 3871–3903. https://doi.org/10.1007/s10664-019-09720-x

APPENDIX

A MODELS FOR ESTIMATING THE IMPACT OF STALE BOT

Table 6. Models for estimating the impact of adopting Stale bot on the number of merged PRs (*merged_pulls*) and closed PRs (*closed_pulls*) in the studied projects.

	log(merged_pulls)		**log(closed_pulls)**	
	Coefficient	**Sum Sq.**	**Coefficient**	**Sum Sq.**
time	0.041***	4.852***	0.038***	4.159***
adoption	-0.129*	0.496*	0.187*	1.040*
time_since_adoption	-0.021*	0.653*	-0.044**	2.721**
log(age_at_adoption)	-0.627**	1.397**	-0.678**	2.483**
log(pulls_at_adoption)	0.883***	1.792***	0.279	0.271
log(contributors_at_adoption)	0.152	0.334	0.379**	3.158**
log(maintainers_at_adoption)	-0.210	0.124	0.226	0.216
intercept	-0.688		0.299	
Marginal R^2	0.57		0.44	
Conditional R^2	0.82		0.70	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 7. Models for estimating the impact of adopting Stale bot on the first response latency of merged PRs (*first_latency_m*) and closed PRs (*first_latency_c*) in the studied projects.

	log(first_latency_m)		**log(first_latency_c)**	
	Coefficient	**Sum Sq.**	**Coefficient**	**Sum Sq.**
time	0.029*	2.398*	0.085***	20.577***
adoption	0.064	0.123	0.071	0.151
time_since_adoption	-0.045*	2.920*	-0.051	3.708
log(age_at_adoption)	0.983	0.786	1.162	2.381
log(pulls_at_adoption)	0.662	0.230	1.342	2.050
log(contributors_at_adoption)	-0.228	0.173	-0.748	4.027
log(maintainers_at_adoption)	-0.950	0.577	-1.257	2.186
intercept	-2.599		-4.939	
Marginal R^2	0.08		0.14	
Conditional R^2	0.87		0.77	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 8. Models for estimating the impact of adopting Stale bot on the mean response latency of merged PRs (*mean_latency_m*) and closed PRs (*mean_latency_c*) in the studied projects.

	log(mean_latency_m)		**log(mean_latency_c)**	
	Coefficient	**Sum Sq.**	**Coefficient**	**Sum Sq.**
time	0.011	0.318	0.031	2.728
adoption	-0.023	0.015	0.128	0.486
time_since_adoption	-0.006	0.052	-0.025	0.901
log(age_at_adoption)	0.677*	1.240*	0.770	2.698
log(pulls_at_adoption)	0.133	0.031	0.467	0.640
log(contributors_at_adoption)	0.093	0.095	-0.131	0.319
log(maintainers_at_adoption)	-0.274	0.160	-0.504	0.907
intercept	-0.158		-0.142	
Marginal R^2	0.22		0.15	
Conditional R^2	0.72		0.58	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 9. Models for estimating the impact of adopting Stale bot on the resolution time of merged PRs (*resolution_time_m*) and closed PRs (*resolution_time_c*) in the studied projects.

	log(resolution_time_m)		**log(resolution_time_c)**	
	Coefficient	**Sum Sq.**	**Coefficient**	**Sum Sq.**
time	-0.001	0.004	0.029*	2.366*
adoption	0.052	0.080	0.267	2.124
time_since_adoption	-0.001	0.001	-0.043*	2.602*
log(age_at_adoption)	0.509	0.460	0.547	1.007
log(pulls_at_adoption)	-0.345	0.136	-0.029	0.002
log(contributors_at_adoption)	0.228	0.376	0.156	0.336
log(maintainers_at_adoption)	0.333	0.154	0.102	0.027
intercept	3.482		3.021	
Marginal R^2	0.25		0.18	
Conditional R^2	0.80		0.66	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 10. Models for estimating the impact of adopting Stale bot on the number of comments in merged PRs (*comments_m*) and closed PRs (*comments_c*) in the studied projects.

	log(comments_m)		**log(comments_c)**	
	Coefficient	**Sum Sq.**	**Coefficient**	**Sum Sq.**
time	-0.009	0.216	-0.008	0.195
adoption	0.023	0.016	-0.001	0.000
time_since_adoption	0.001	0.002	-0.003	0.015
log(age_at_adoption)	-0.165	0.020	-0.075	0.009
log(pulls_at_adoption)	-0.603	0.171	-0.438	0.209
log(contributors_at_adoption)	0.221	0.145	0.300*	0.617*
log(maintainers_at_adoption)	0.826*	0.390*	0.375	0.186
intercept	3.252		2.589	
Marginal R^2	0.26		0.19	
Conditional R^2	0.90		0.79	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 11. Models for estimating the impact of adopting Stale bot on the number of commits in merged PRs (*commits_m*) and closed PRs (*commits_c*) in the studied projects.

	log(commits_m)		**log(commits_c)**	
	Coefficient	**Sum Sq.**	**Coefficient**	**Sum Sq.**
time	0.010	0.271	0.030*	2.501*
adoption	-0.021	0.013	-0.215	1.369
time_since_adoption	-0.015*	0.335*	-0.026	0.942
log(age_at_adoption)	0.189	0.094	0.545**	7.487**
log(pulls_at_adoption)	-0.252	0.108	-0.120	0.234
log(contributors_at_adoption)	-0.019	0.004	0.085	0.741
log(maintainers_at_adoption)	0.064	0.009	-0.188	0.700
intercept	2.580		0.783	
Marginal R^2	0.09		0.16	
Conditional R^2	0.67		0.27	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 12. Models for estimating the impact of adopting Stale bot on the number of active contributors (*contributors*) in the studied projects.

	log(contributors)	
	Coefficient	**Sum Sq.**
time	0.026***	1.867***
adoption	0.003	0.000
time_since_adoption	-0.022***	0.708***
log(age_at_adoption)	-0.535**	0.493**
log(pulls_at_adoption)	0.250	0.070
log(contributors_at_adoption)	0.697***	3.411***
log(maintainers_at_adoption)	0.114	0.018
intercept	-1.033	
Marginal R^2	0.84	
Conditional R^2	0.96	

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

B VARIATION OF THE PERFORMANCE INDICATORS

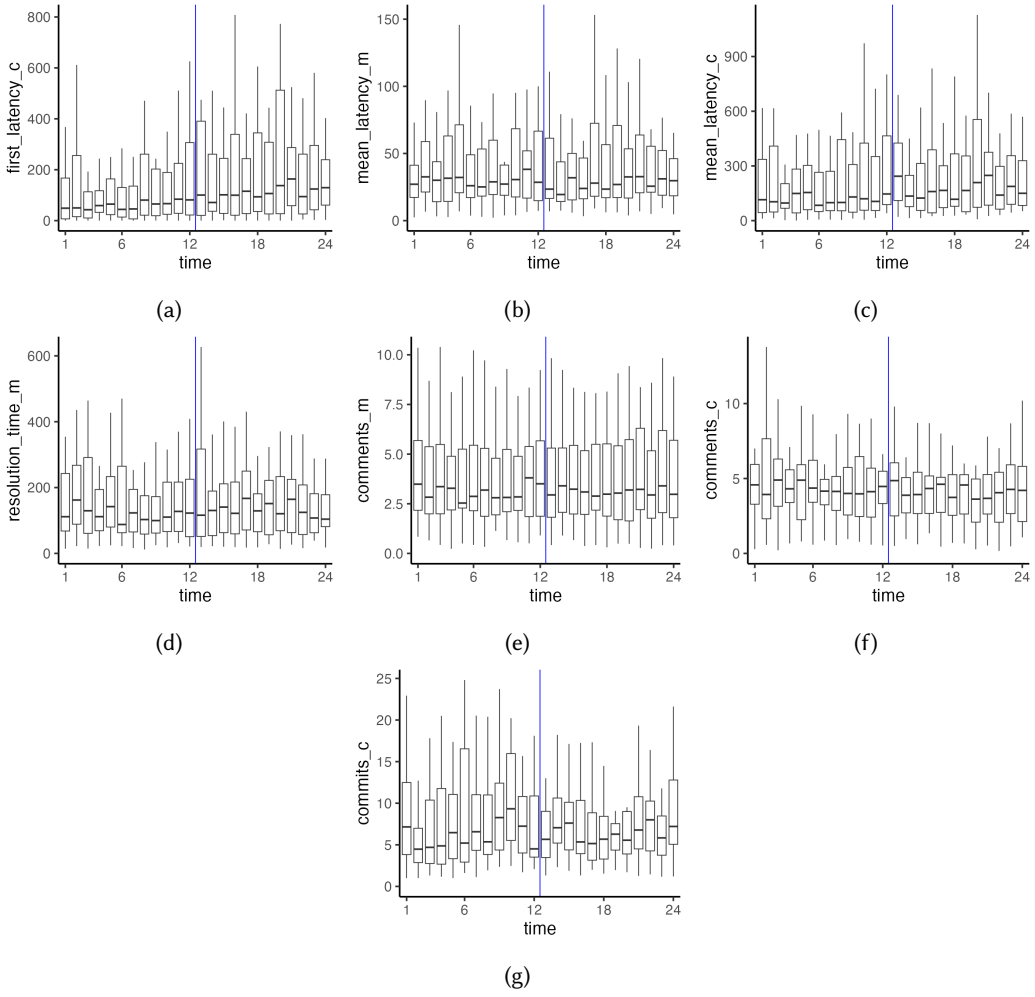


Fig. 6. Variation in (a) the first response latency of closed PRs, (b) the mean response latency of merged PRs, (c) the mean response latency of closed PRs, (d) the resolution time of merged PRs, (e) the number of comments in merged PRs, (f) the number of comments in closed PRs, and (g) the number of commits in closed PRs each month during our observation period. The blue lines show the adoption time.

C CHARACTERISTICS OF PRS INTERVENED BY STALE BOT

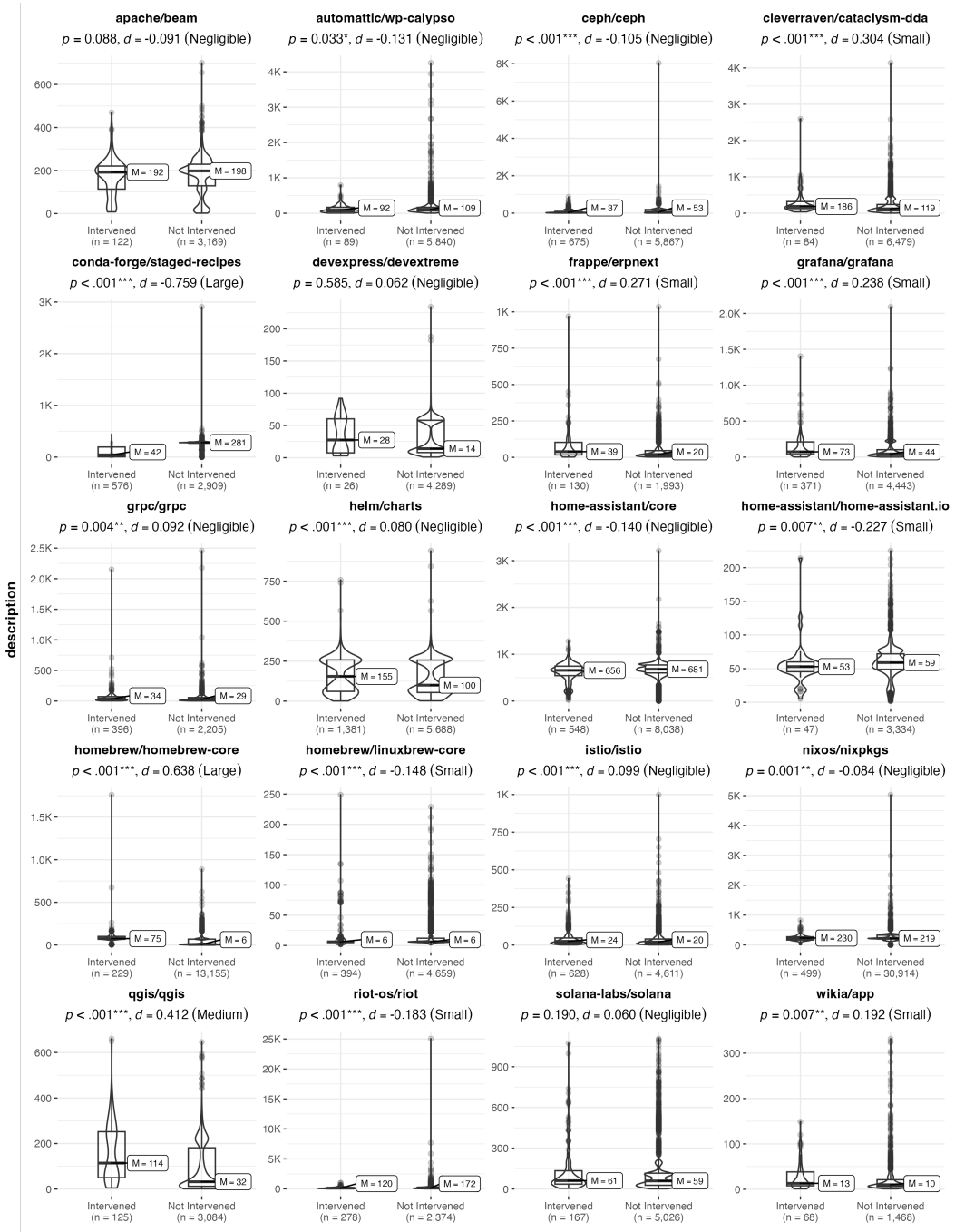


Fig. 7. Comparison of intervened and not intervened PRs regarding their description length across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

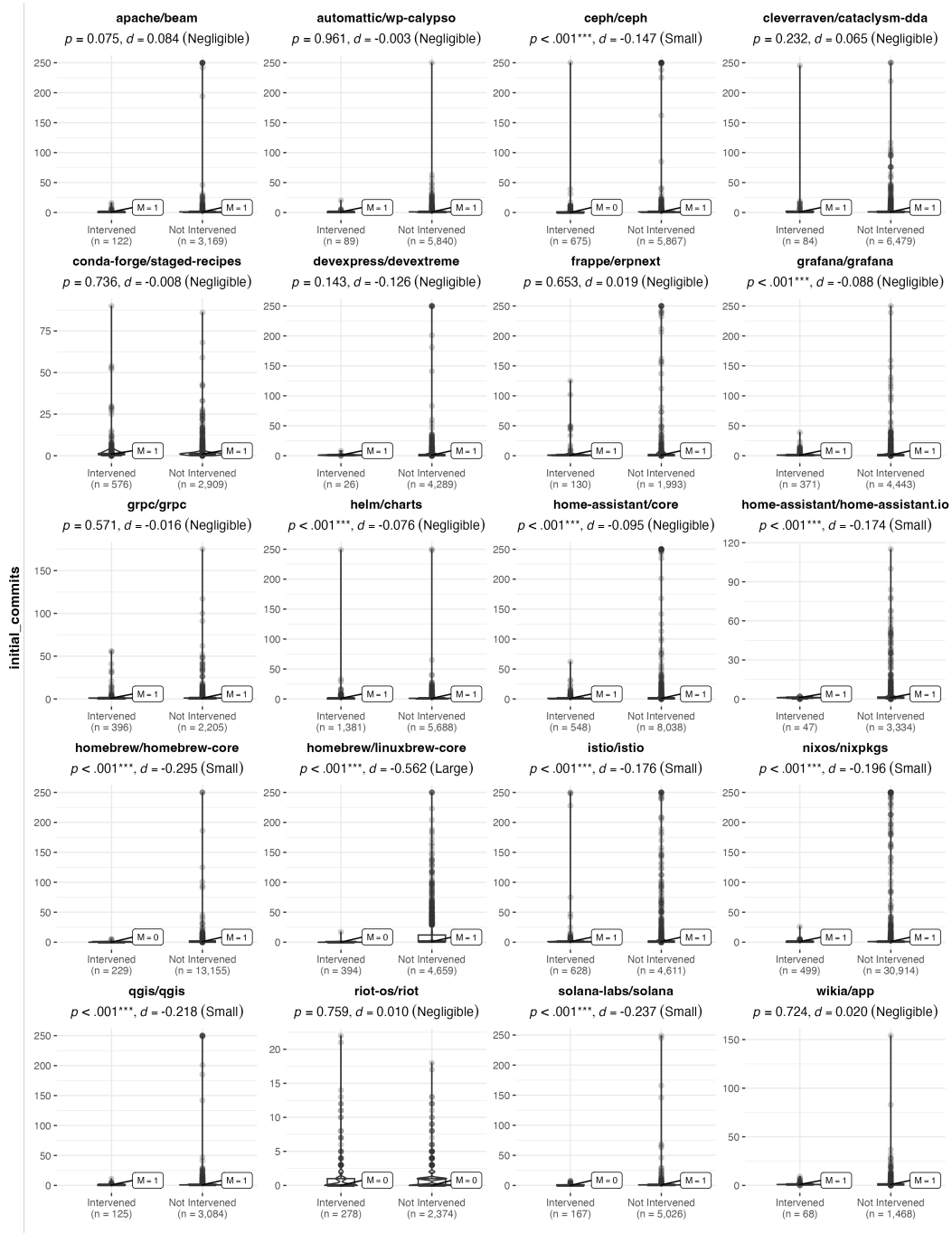


Fig. 8. Comparison of intervened and not intervened PRs regarding their number of initial commits across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

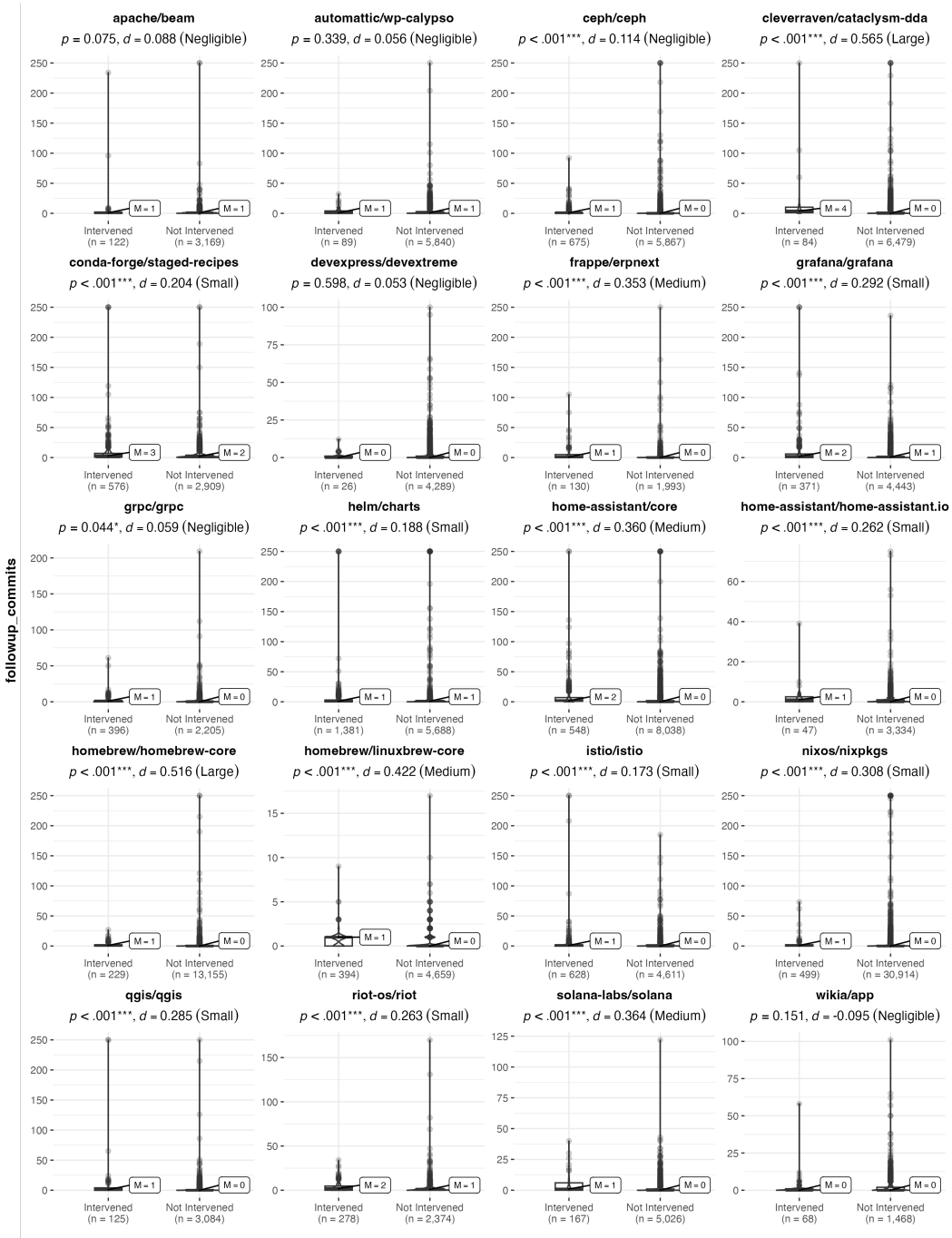


Fig. 9. Comparison of intervened and not intervened PRs regarding their number of follow-up commits across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

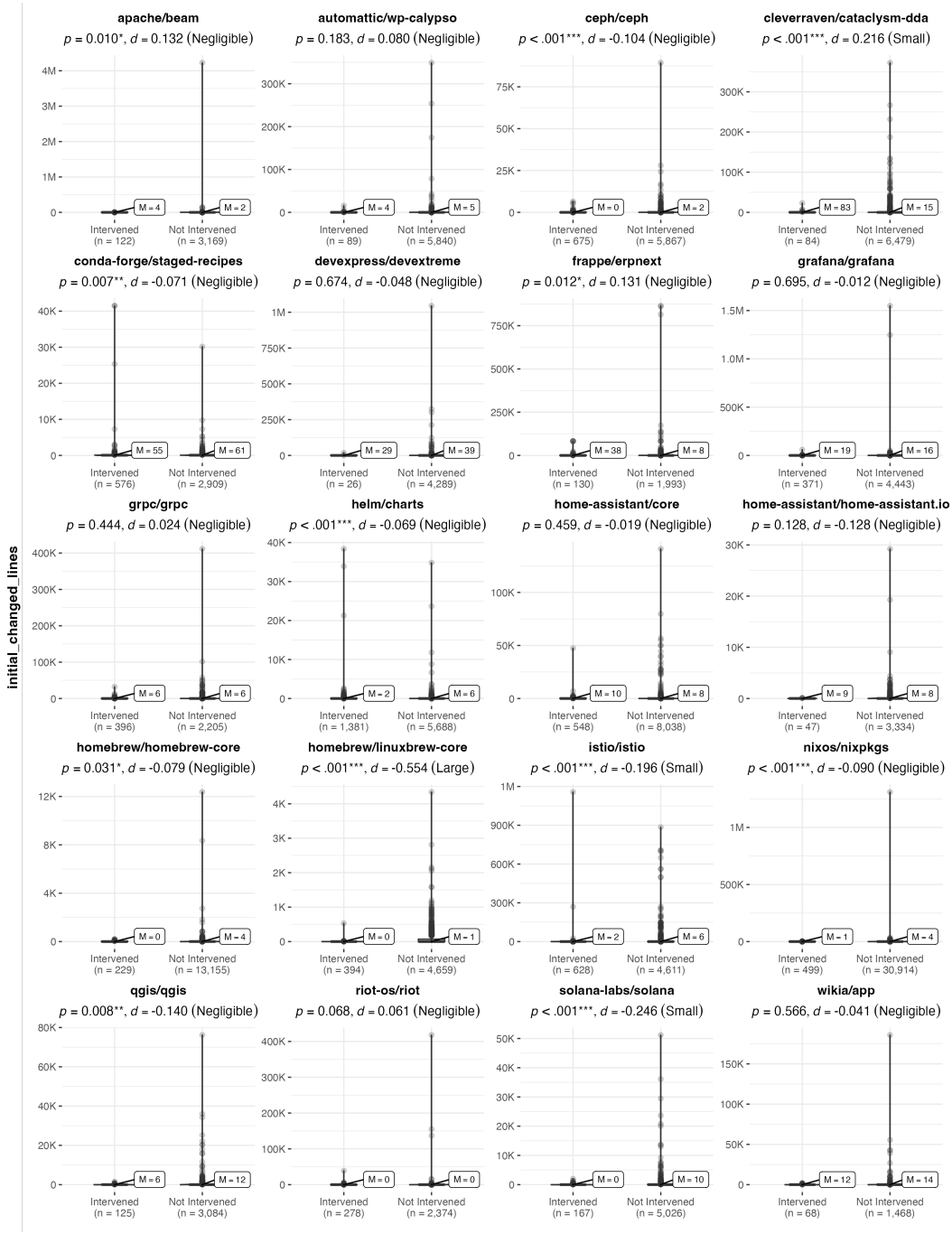


Fig. 10. Comparison of intervened and not intervened PRs regarding their number of initial changed lines across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

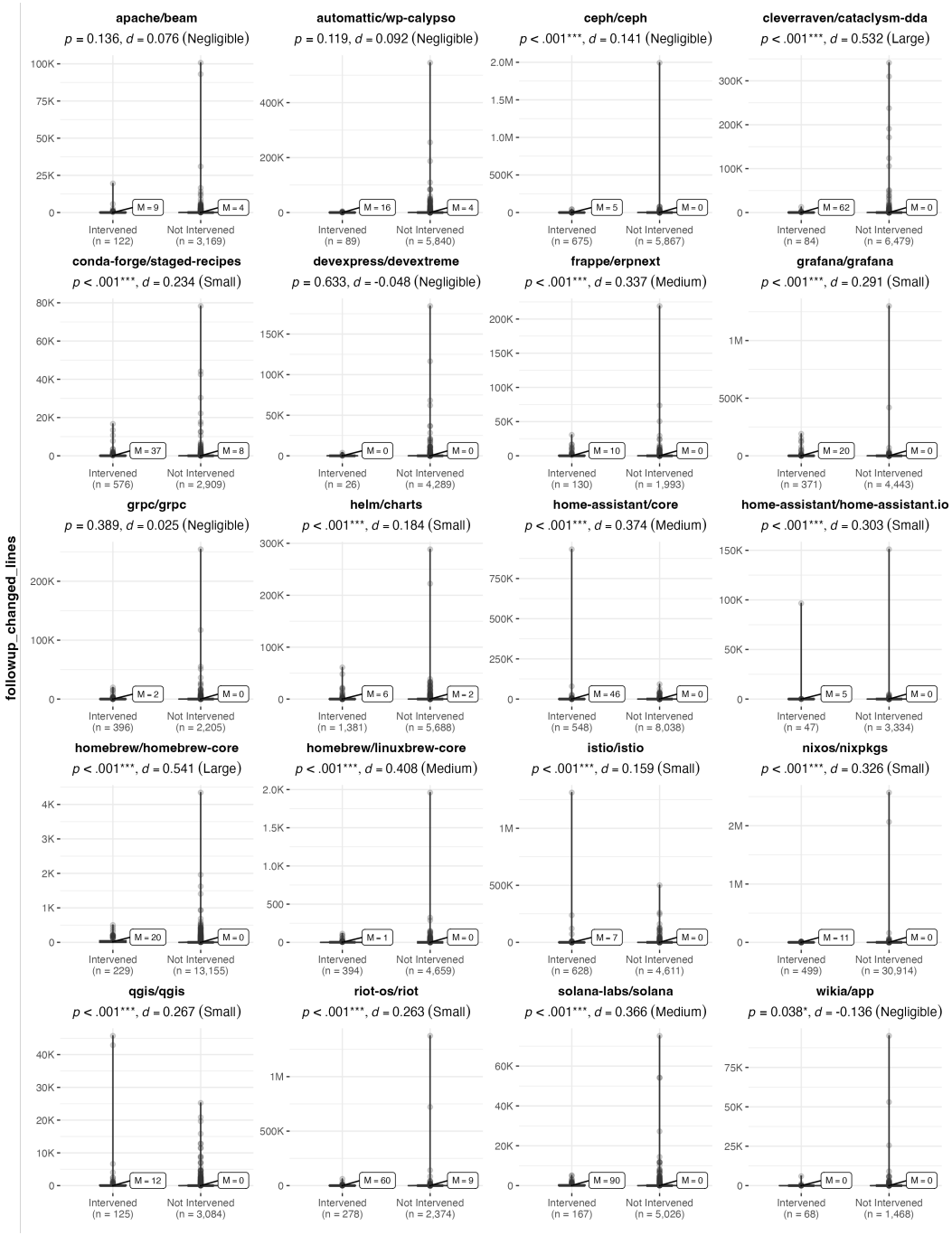


Fig. 11. Comparison of intervened and not intervened PRs regarding their number of follow-up changed lines across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

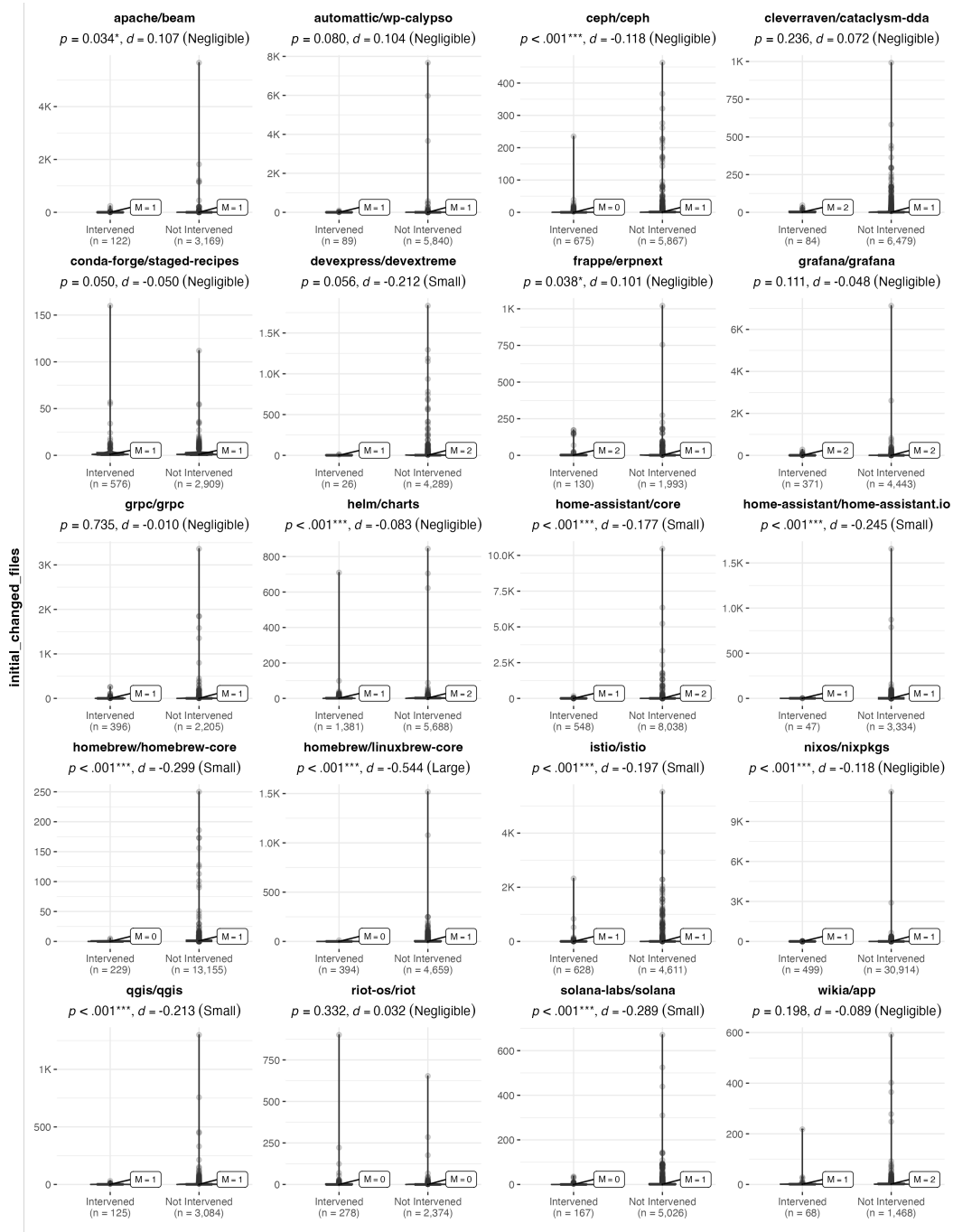


Fig. 12. Comparison of intervened and not intervened PRs regarding their number of initial changed files across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

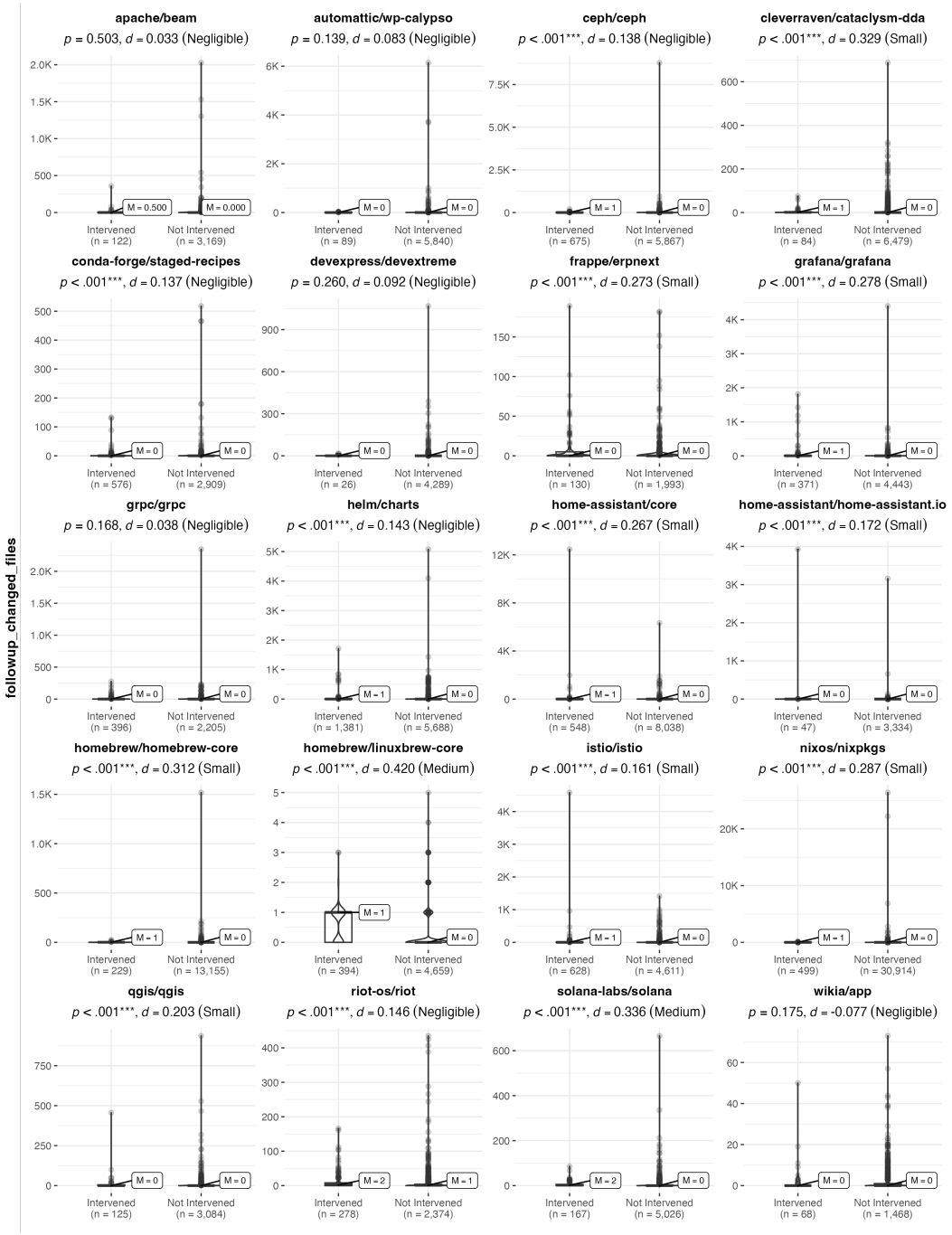


Fig. 13. Comparison of intervened and not intervened PRs regarding their number of follow-up changed files across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

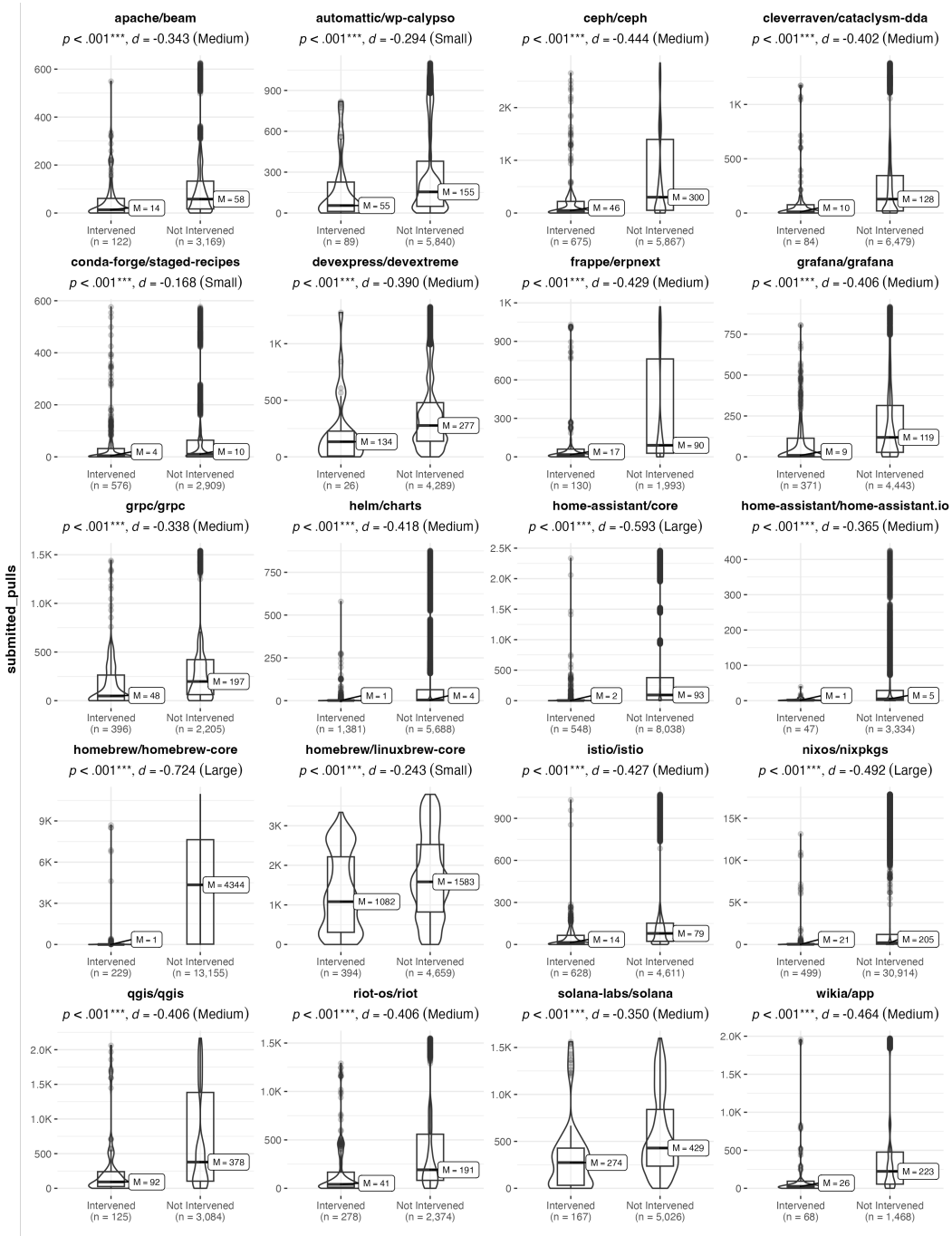


Fig. 14. Comparison of intervened and not intervened PRs regarding the number of prior PRs by their contributors across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

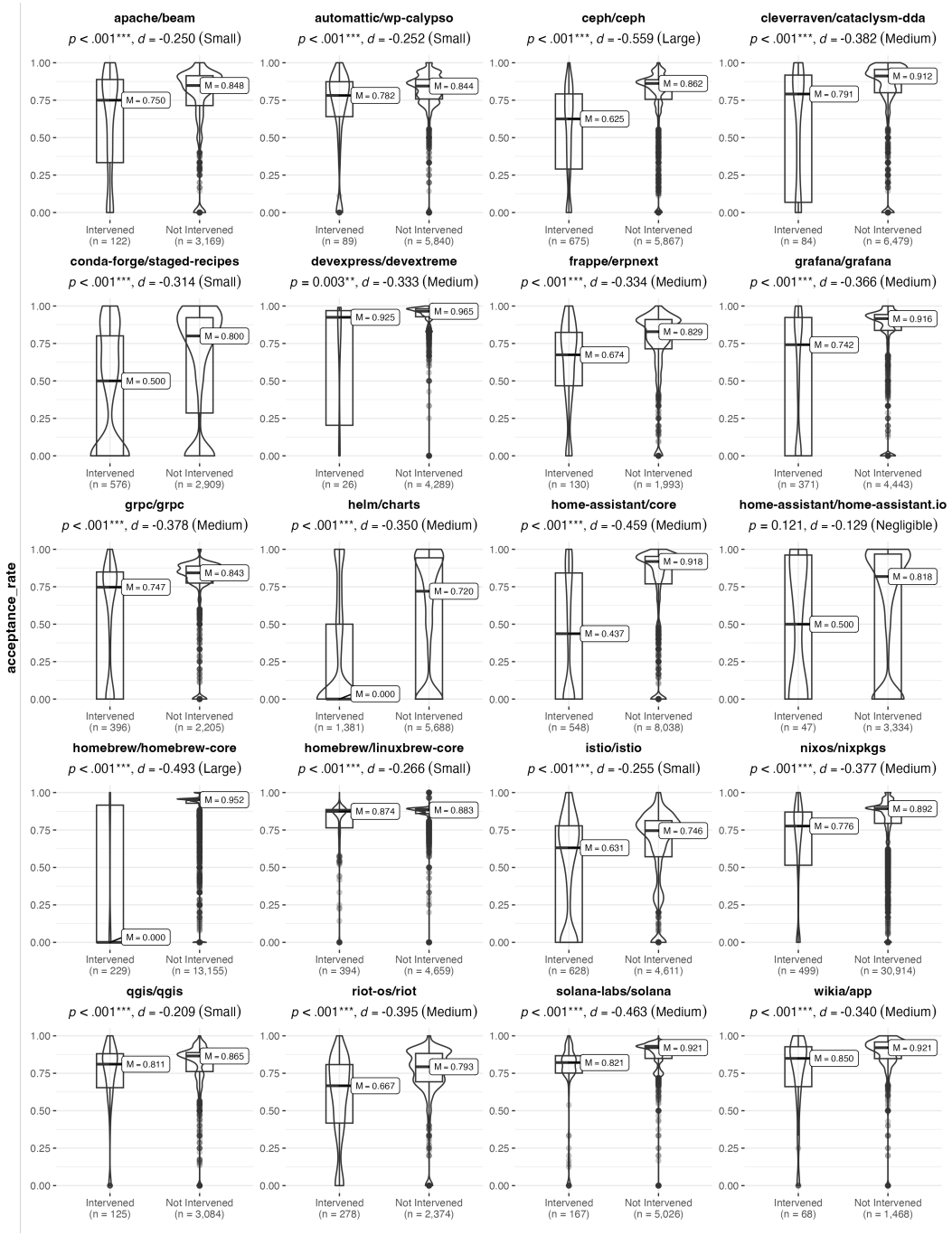


Fig. 15. Comparison of intervened and not intervened PRs regarding the acceptance rate of their contributors across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

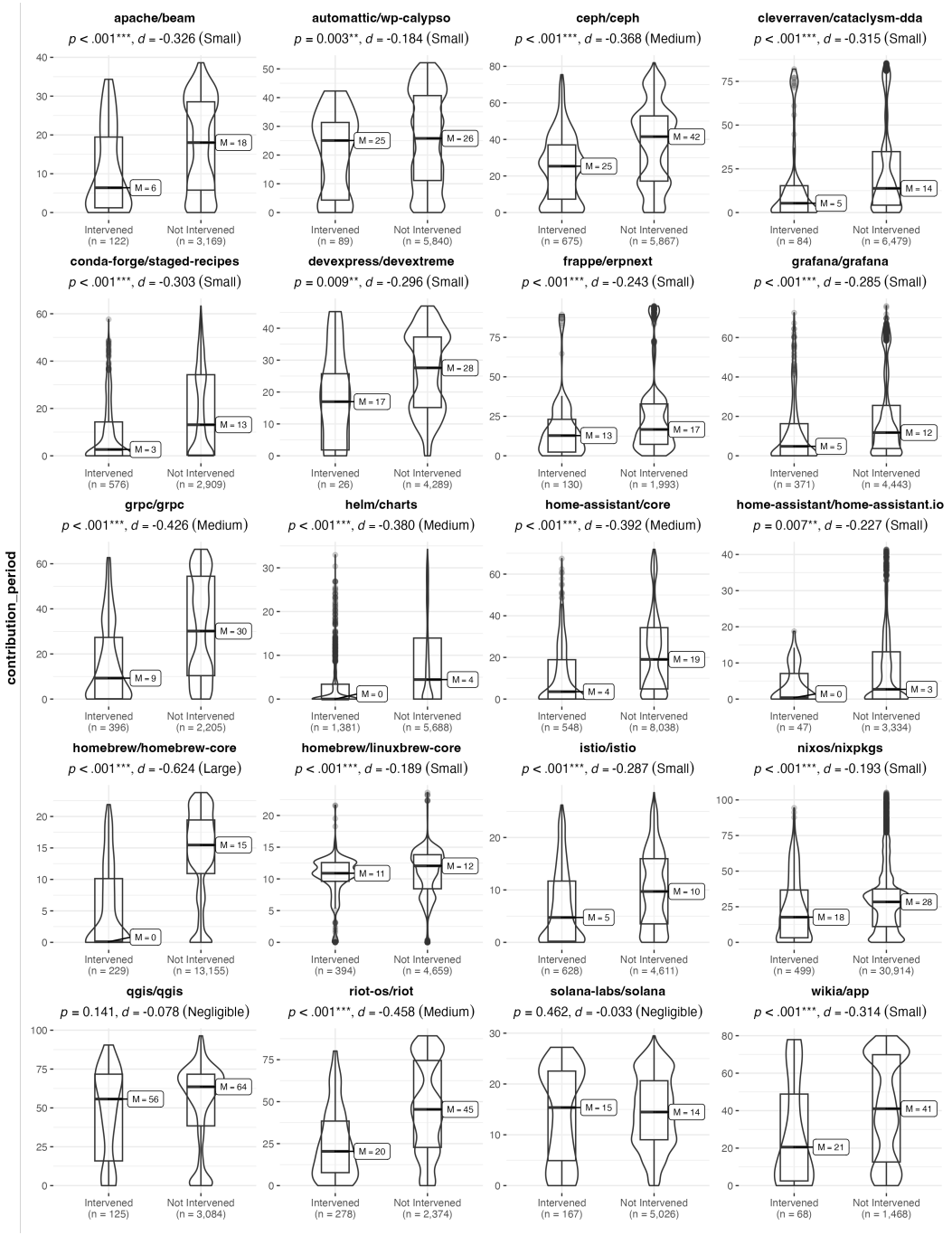


Fig. 16. Comparison of intervened and not intervened PRs regarding the contribution period of their contributors across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

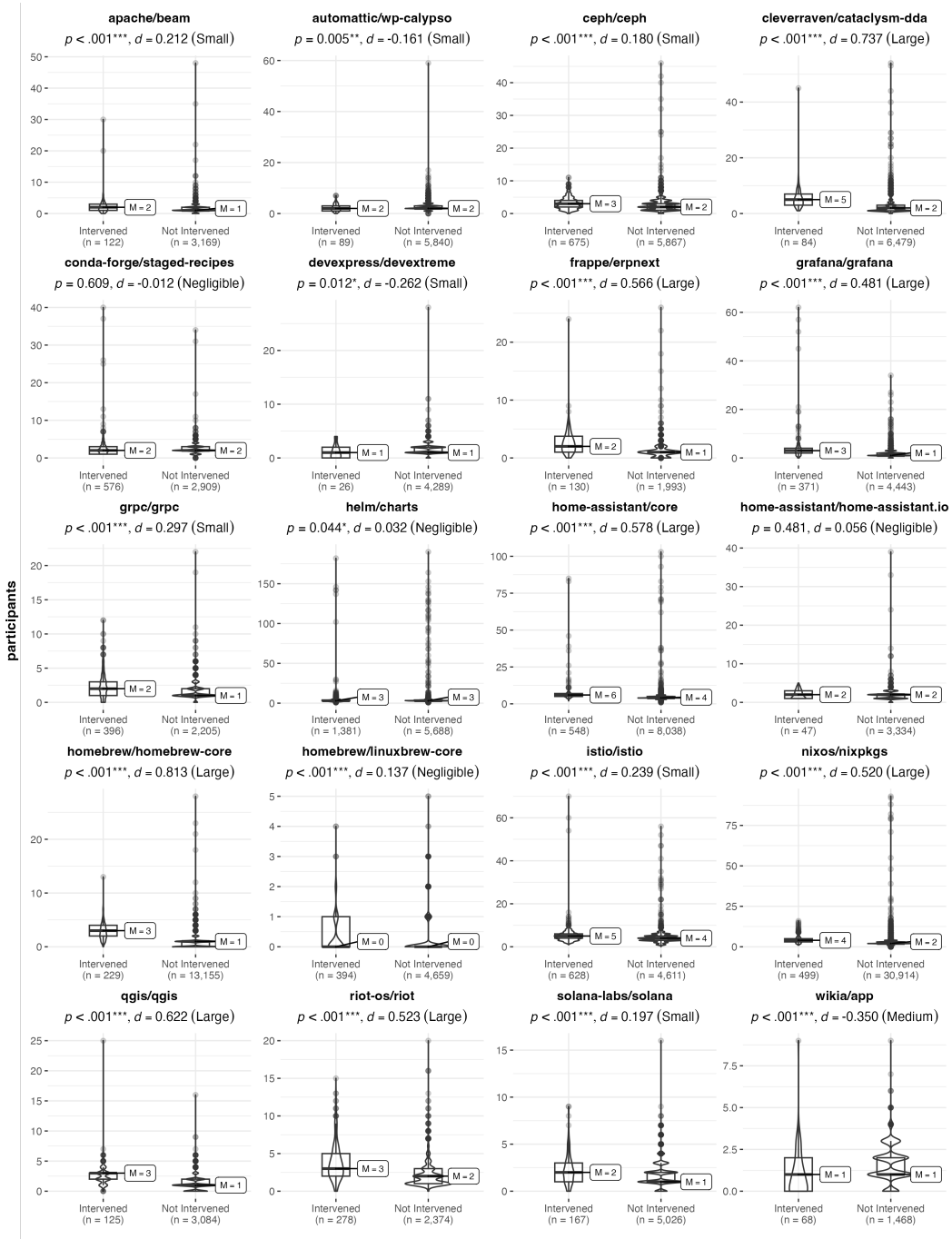


Fig. 17. Comparison of intervened and not intervened PRs regarding the number of participants in their review process across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

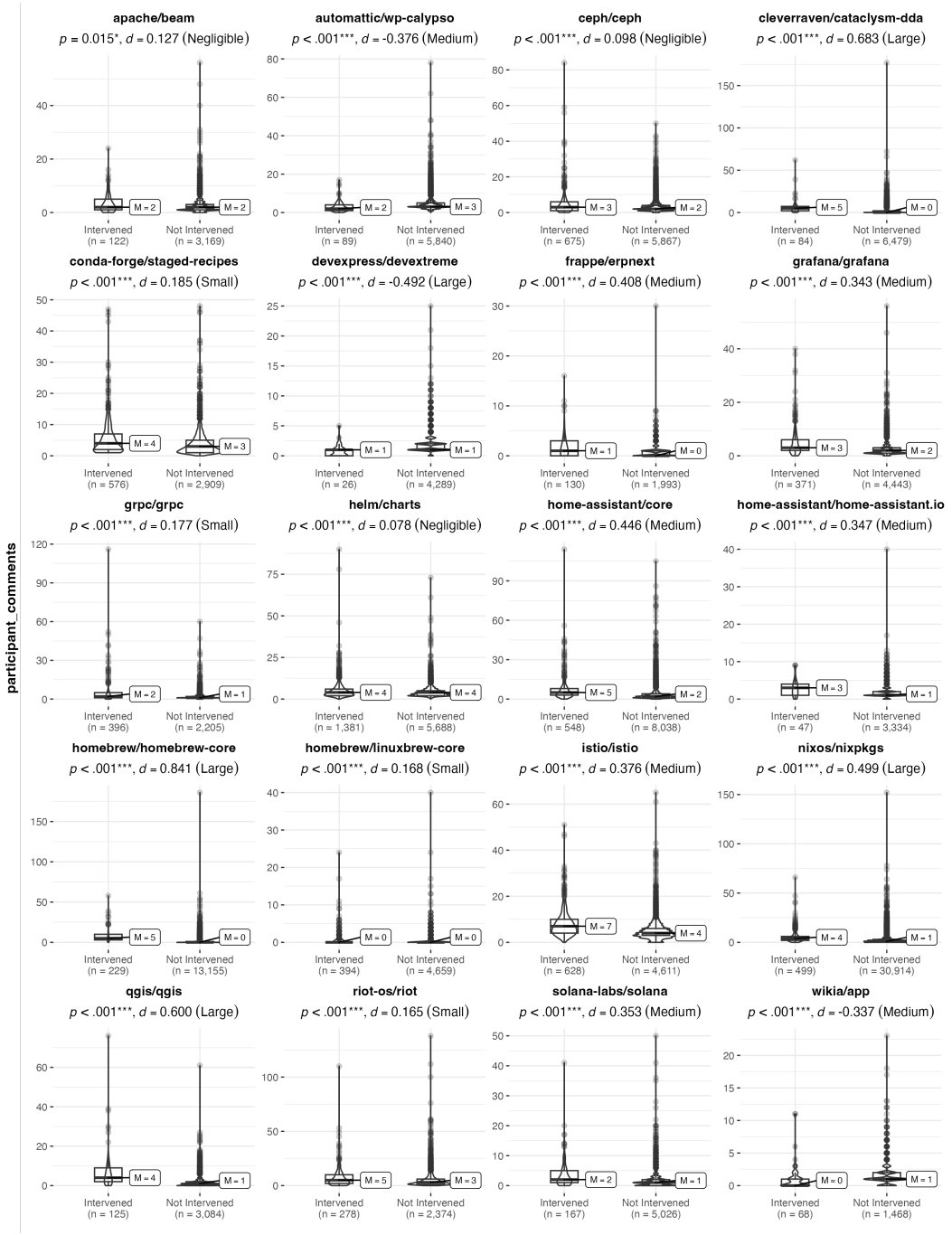


Fig. 18. Comparison of intervened and not intervened PRs regarding the number of participant comments in their review process across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

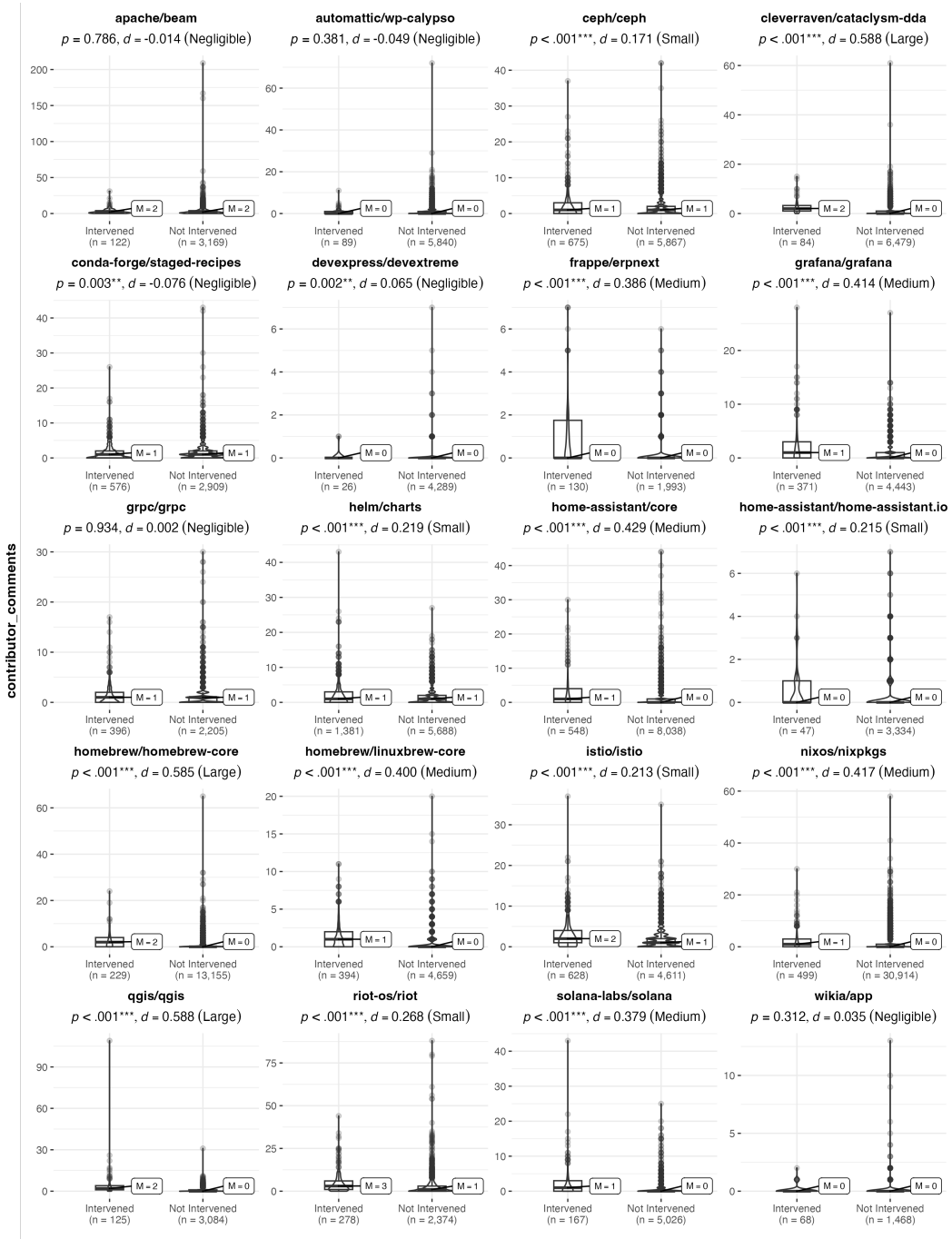


Fig. 19. Comparison of intervened and not intervened PRs regarding the number of contributor comments in their review process across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

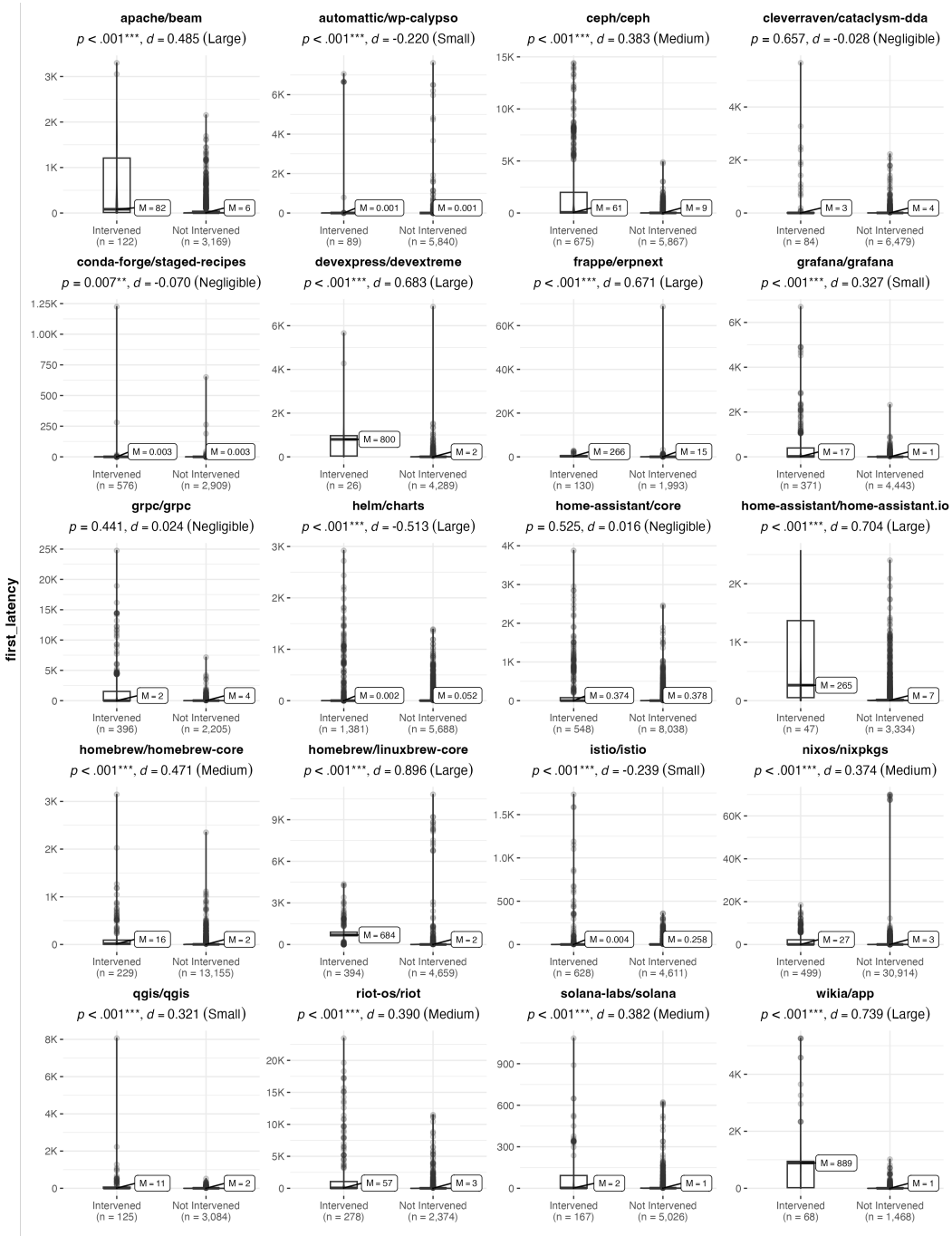


Fig. 20. Comparison of intervened and not intervened PRs regarding their first response latency across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

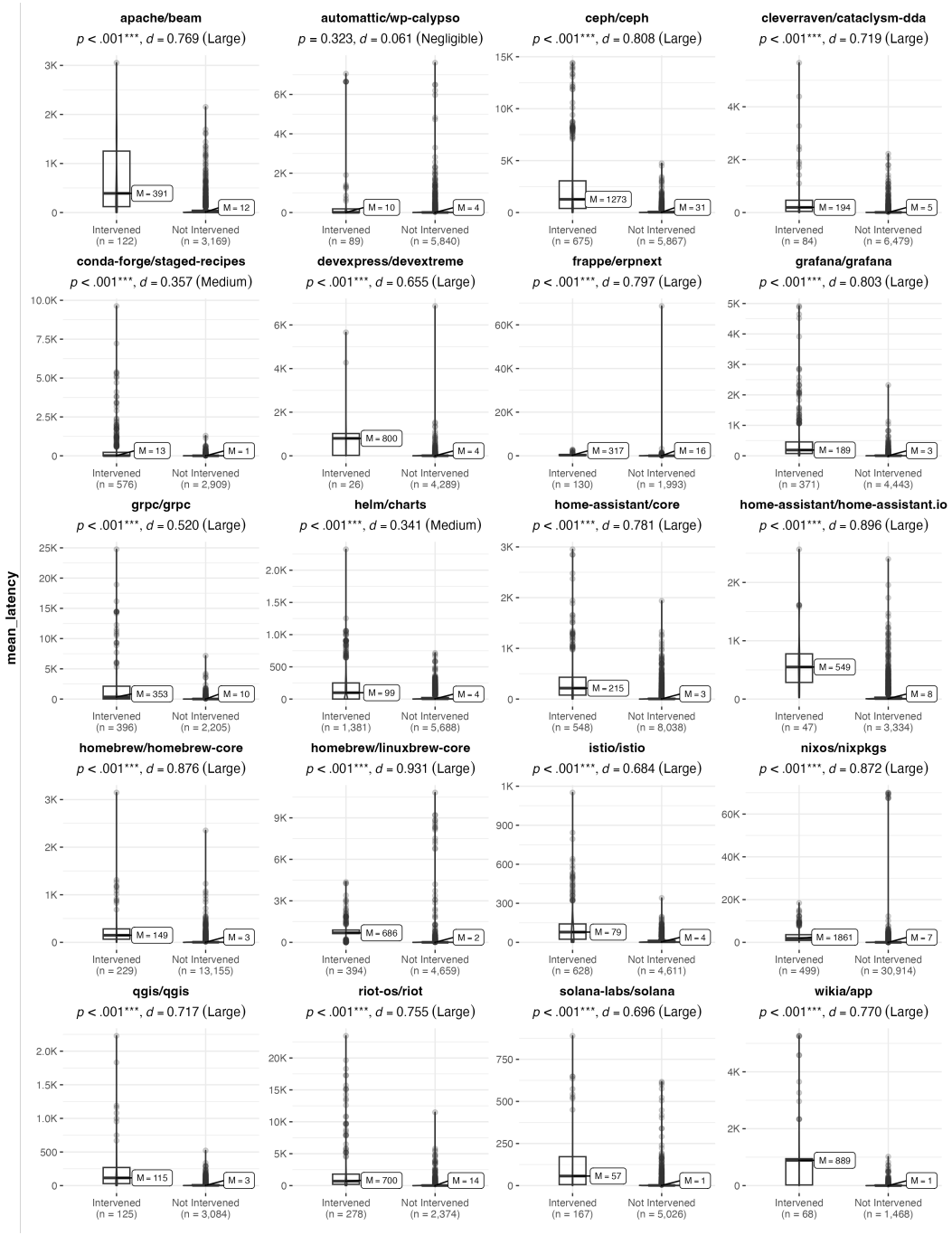


Fig. 21. Comparison of intervened and not intervened PRs regarding their mean response latency across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

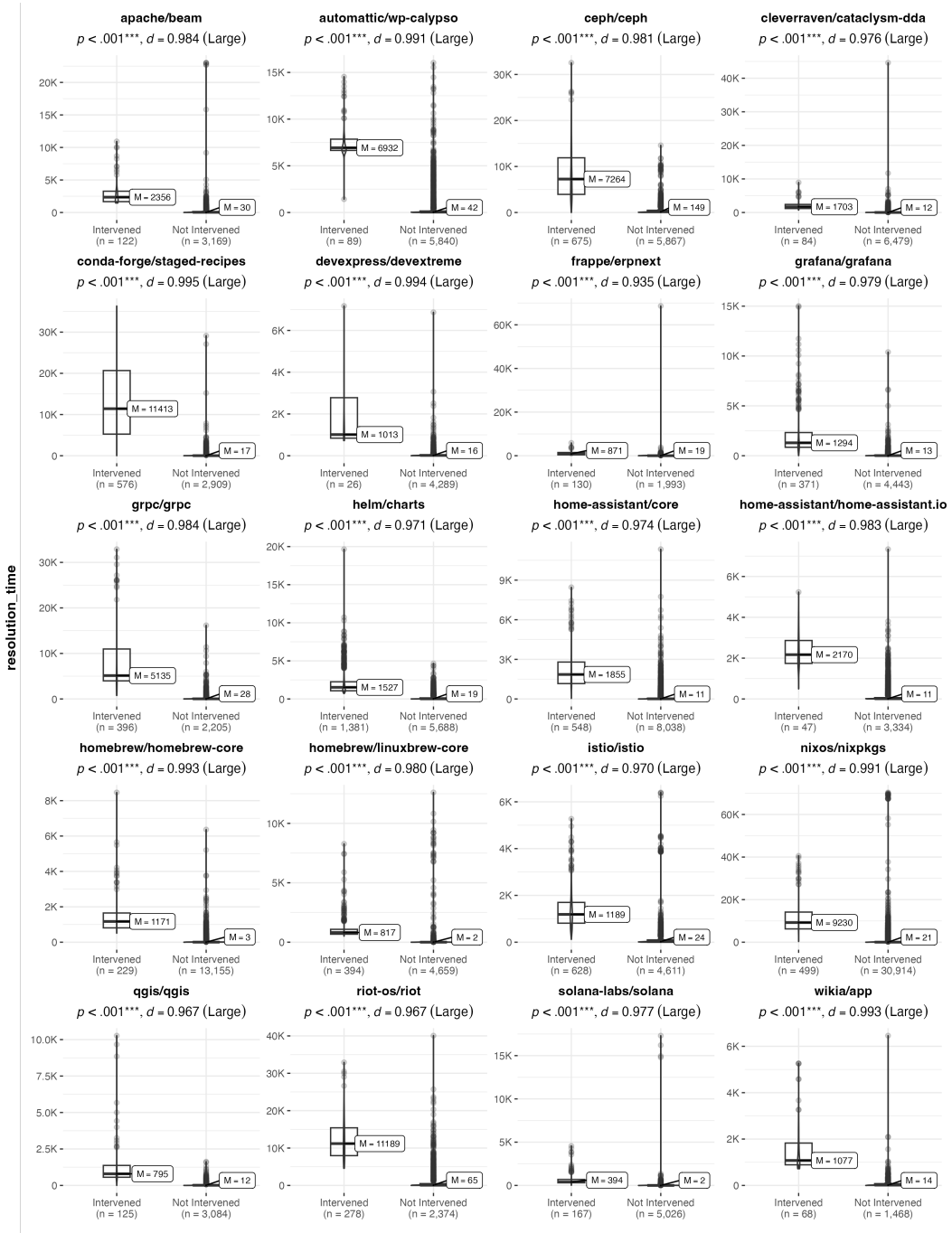


Fig. 22. Comparison of intervened and not intervened PRs regarding their resolution time across the studied projects. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$