Code Contribution and Scientific Authorship

Eva Maxfield Brown[a,*], Isaac Slaughter[a], Nicholas Weber[a]

[a] *University of Washington Information School,*

Abstract

Software development and scientific collaboration are fundamental aspects of contemporary research, yet quantitative science studies typically examine them separately. We develop a dataset of approximately 140,000 paired research articles and code repositories, and a predictive model that matches research article authors with software repository developer accounts. With these resources, we bridge the two literatures - invetigating how software development activities influence credit allocation in collaborative scientific settings. Our findings reveal significant patterns distinguishing software contributions from traditional authorship credit. Nearly 30% of articles include non-author code contributors—individuals who participated in software development but received no formal authorship recognition. While code-contributing authors show a modest 5.1% increase in article citations, this effect becomes non-significant when controlling for domain, article type, and open access status. First authors are significantly more likely to be code contributors than other author positions. Notably, we identify a negative relationship between coding frequency and scholarly impact metrics. Authors who contribute code more frequently exhibit progressively lower h-indices compared to non-coding colleagues, even when controlling for publication count, author position, domain, and article type. These results suggest a disconnect between software contributions and conventional academic recognition systems, highlighting important implications for scientific reward structures in increasingly computational research environments.

1. Introduction

Software and collaboration are two essential ingredients in contemporary science. Quantaitive studies of science tend to explore these two topics separately: Collaboration is often studied using digital trace data or biblimetric metadata to understand, for example, who works with whom (Newman et al., 2014), recieves what credit (Shen and Barabási, 2014), and to what effect (Ahuja, 2000). Similarly, scientific software is often quantatively studied by extracting mentions (Du et al., 2021) or formal citations (Du et al., 2022) that connect scientists and software to specific impact metrics (i.e. citation counts)(Park and Wolfram, 2019).

In the following paper we unite these two literatures to better understand how the development of software effects the allocation of credit in collaborative settings. In doing so, we take advantage of advances in natural language processing and machine learning to develop unique data sources that compliment traditional biblimetric indicators. We first build a comprehensive dataset of ~140,000 paired research articles and code repositories - giving us a verified connection between the software that supports new knowledge claims made in the published literature. Next, we develop a predictive model that matches the authors of a research article and the developer accounts of a linked source code repository - giving us a unique way to understand the allocation of credit for software contributions. We apply the predictive model to our dataset and then use the commit history of public software repositories to test three hypothesis.

[*] Corresponding author
Email address: `evamxb@uw.edu` (Eva Maxfield Brown)

We identify several patterns that distinguish code contributions from formal scientific recognition. We show that ~30% (n=6529) of articles have non-author code-contributors - individuals who may have helped create the software but received no formal authorship credit. We find that code-contributing authors are associated with a modest increase in article-level impact metrics (~5.1% increase in citations per code-contributing author), but these effects become statistically non-significant when controlling for domain, article type, and open access status. First authors are significantly more likely to be code contributors than other author positions across all conditions tested. We also find a negative relationship between coding frequency and scholarly impact: Authors who contribute code more frequently show progressively lower h-indices than their non-coding peers, a pattern that persists when controlling for publication count, and author's most common author position, domain, and article type.

The remainder of this paper proceeds as follows. First, we review related work regarding scientific software development and the recognition of code contributors in scientific credit systems. In doing so, we motive three specific hypotheses that guide our investigation. Next, we detail our data and methods, describing how we linked article-repository pairs, trained and evaluated a predictive model for entity matching, and applied this model across each article-repository pair. We then present our analysis, focusing on article-level dynamics before moving to individual-level patterns, formally accepting or rejecting each hypothesis based on our findings. We conclude by discussing the results, limitations of our work, and areas for future improvement.

2. Background

Software and collaboration are two essential ingredients for contemporary science. We need look no further than two recent Nobel Prizes to confirm this idea: In Biology, Alphafold enabled the large-scale prediction of novel protein structures and was the main contribution of the 2024 Nobel Prize winners in Chemistry (Jumper et al., 2021). In Physics, LIGO's development of numerous open-source tools, including GstLAL, enabled the first detection of a gravitational wave and the 2017 Nobel Prize for three of the project's directors (Cannon et al., 2021). These two Nobel cases also demonstrate an essential tension between scientific software and collaboration: Recognition is not distributed evenly. A large number of people develop a tool, but only a few are able to use and be rewarded for the findings it produced.

The proper allocation of credit has been a longstanding topic in social studies of science. Merton's description of a "Matthew Effect" is perhaps the most famous account of a cumulative advantage in publishing - established scientists often receive more attention for work of similar importance or value than less established peers (Merton, 1968). In quantitative work, credit is most often established using a proxy measure from bibliographic data like author order, which is an imperfect means of assigning credit and doing so often exacerbates existing inequalities (West et al., 2013). More recent work developing algorithms to assign credit (Shen and Barabási, 2014), and to formally document contributor roles (Allen et al., 2014) are aimed at trying to improve upon author order as the status quo mechanism for assigning credit.

Recognition of technical contributions to scientific work is also a longstanding topic in social studies of science. Shapin's history of innovation describes how "invisible technicians" allowed Francis Bacon and Robert Boyle to, as the latter describes, "do experiments by others hands" (Shapin, 1989). As a technical artifact, scientific software is recognized as being important to contemporary research (Hettrick et al., 2014) but many existing accounts of devlopment and long-term maintenance describe inequalities in credit, recognition, and career advancement (Muna et al., 2016). Much of the existing literature on the role and importance of software in science is ethnographic (Howison and Herbsleb, 2011, Paine and Lee (2017), Neang et al. (2023), Hannay et al. (2009)) or small scale surveys (Carver et al., 2022, Momcheva and Tollerud (2015)).

These two literatures - collaboration and scientific software - are rarely connected via quantiative research. In this paper, we build unique datasets that allow us to unite these two bodies of parallel findings on the allocation of credit in collaborative settings, and the development of scientific software. In doing so, we state three formal hypothesis that we will test in the following sections.

2.1. H1: Research Team Composition and Scientific Attention

In collaborative settings, experimental and theoretical research tends to receive more citations than methods-focused contributions, except when methods represent foundational shifts in a field(Aksnes, 2006; Liu et al., 2023; Chen et al., 2024). Software is often positioned as a methodological contribution and as a result can be found in some of the highest cited papers of the 21st century (Jin et al., 2015; Hampton et al., 2013; Hasselbring et al., 2024).

Prior work also establishes a positive relationship between team size and citation count where larger and more diverse teams typically produce higher-impact scientific work (Franceschet and Costantini, 2010; Larivière et al., 2014). A similar finding in empirical software studies is that larger development teams tend to create more reliable software with fewer defects (Wyss et al., 2023, Herbsleb and Mockus (2003)) though this comes at the expense of slower development cycles. These findings suggest that team size may be particularly important in scientific software development, where technical robustness and scientific innovation are crucial (Milewicz and Raybourn, 2018).

Finally, we argue that the unique characteristics of scientific software development - including implementing novel algorithms, requiring deep domain knowledge, and an increased emphasis on reproducibility (Muna et al., 2016; Howison and Herbsleb, 2013) - make team composition especially relevant. Development in organized teams may enhance scientific impact through multiple mechanisms: they can produce more robust and generalizable software tools for methodological contributions while enabling more sophisticated computational analyses and larger-scale data processing for experimental work.

Given these patterns in team dynamics, software development practices, and the computational transformation of scientific work, we propose:

H1: The number of individuals contributing code to a publication's associated repository positively correlates with the article's citation count.

2.2. H2: Author Roles and Technical Contributions

Author positions in scientific publications signal specific roles and responsibilities (Shen and Barabási, 2014), a relationship extensively studied through contribution role taxonomies like CRediT (Larivière et al., 2016). These studies reveal that first authors and corresponding authors, while occasionally the same individual (Chinchilla-Rodríguez et al., 2022), take on distinct responsibilities. Analyses of contribution patterns consistently show that software development, data analysis, and visualization tasks typically fall to first authors (Larivière et al., 2016; Júnior et al., 2016; Larivière et al., 2020; Sauermann and Haeussler, 2017). Meanwhile, corresponding authors, whether or not they are also first authors, often maintain responsibility for research artifacts' long-term sustainability and reuse, which we believe may include the maintenance and documentation of software tools.

Contribution records from source code repositories provide a unique method to verify these established contribution patterns. Given prior findings about the distribution of technical responsibilities within research teams, we expect these repository records to reflect similar patterns of engagement with software development:

H2a: First authors have higher code contribution rates than authors in other positions.

H2b: Corresponding authors have higher code contribution rates than non-corresponding authors.

2.3. H3: Code Contribution and Individual Scientific Impact

Despite the increasingly central role of software in science, researchers who develop scientific software face persistent challenges in receiving formal scientific recognition for their contributions. Prior work has shown that software developers in research settings are often relegated to acknowledgment sections rather than receiving authorship credit, even when their technical contributions are fundamental to the research (Carver et al., 2022; Philippe et al., 2019).

The challenge of recognition is compounded by inconsistent practices in software citation. While researchers have tried to standardize software citation, actual citation practices remain highly variable across fields and journals (Lamprecht et al., 2020; Katz et al., 2020; Smith et al., 2016). This variability challenges researchers who maintain and update existing software packages. While creating entirely new software may lead to dedicated publications and citations, the ongoing work of maintaining, debugging, and extending existing software - often crucial for scientific progress - typically generates less visible scientific credit (Howison and Herbsleb, 2011, 2013).

These structural challenges in recognizing and citing software contributions suggest a potential misalignment between technical contributions and traditional scientific impact metrics. When researchers dedicate significant time to software development and maintenance, conventional bibliometric measures may not fully capture their contributions, regardless of the software's importance to the field. Whether through attribution practices that favor acknowledgments over authorship or citation patterns that undervalue maintenance work, multiple mechanisms could lead to lower traditional impact metrics for active code contributors. Based on these patterns in software recognition and citation, we hypothesize:

H3: The frequency with which individual researchers contribute code to their research projects is negatively correlated with their h-index.

3. Data and Methods

Our analysis examines the relationship between software development contributions and scientific credit attribution through a three-step process: (1) building a dataset of linked scientific articles and code repositories, (2) developing a predictive model to match article authors with developer accounts, and (3) analyzing patterns in these relationships.

Our dataset integrates article-repository pairs from four sources, each with explicit mechanisms for code sharing: the Journal of Open Source Software (JOSS) and SoftwareX require code repositories as part of publication, Papers with Code directly connects preprints with implementations, and Public Library of Science (PLOS) articles include mandatory data and code availability statements that we mined for repository links. We focused exclusively on GitHub-hosted repositories, which represent the predominant platform for scientific software sharing (Cao et al., 2023; Escamilla et al., 2022). For each article-repository pair, we resolved DOIs via Semantic Scholar to ensure we captured the latest version of each publication, extracted publication metadata and author metrics through OpenAlex, and collected repository information via the GitHub API This created a comprehensive dataset with information about both the scientific and software development aspects of each project. All data was collected between October and November 2024.

To match article authors with repository developer accounts, we developed a machine learning approach using transformer-based architectures. Specifically, we use transformers to overcome entity matching challenges, as exact name or email matching is insufficient due to formatting variations and incomplete information (e.g., "J. Doe" vs. "Jane Doe" in publications, or use of institutional versus personal email addresses). While exact matching fails, there is typically high semantic overlap between author information and developer account details that our model can leverage. We created a gold-standard dataset of 3,000 annotated author-developer account pairs from JOSS publications, where two independent reviewers classified each pair as matching or non-matching. After systematic evaluation of three transformer architectures with various feature combinations, our optimal model (fine-tuned DeBERTa-v3-base including developer account username and display name in the training data) achieved an F1 score of 0.944, with 0.938 precision and 0.95 recall (detailed comparison available in the Appendix).

Applying our model across all article-repository pairs yielded a large-scale dataset linking scientific authors and code contributors. As shown in Table 1, our full, unfiltered dataset contains approximately 140,000 unique article-repository pairs spanning multiple domains and publication types, with nearly 300,000 distinct authors and more than 150,000 developer accounts. From these, we identified almost 110,000 author-developer account relationships, creating a unique resource for investigating code contribution patterns in scientific teams. This dataset enables systematic examination of how software development work relates to scientific recognition and career trajectories. Both the dataset (https://doi.org/10.7910/DVN/KPYVI1) and model (https://huggingface.co/evamxb/dev-author-em-clf) are made publicly available to support further research.

Table 1: Counts of Article-Repository Pairs, Authors, and Developers by Data Sources, Domains, Document Types, and Access Status in the Full, Unfiltered Dataset.

Category	Subset	Article-Repository Pairs	Authors	Developers
By Domain	Health Sciences	5176	26022	7277
	Life Sciences	7727	31613	12123
	Physical Sciences	116597	240536	130601
	Social Sciences	8839	29239	14023
By Document Type	preprint	72177	170301	87311
	research article	63528	173183	78935
	software article	2891	9294	12868
By Access Status	Closed	5740	23668	9352
	Open	132856	286874	147831
By Data Source	joss	2336	7105	11362
	plos	6090	30233	8784
	pwc	129615	262889	134926
	softwarex	555	2244	1628
Total		**138596**	**295806**	**152170**

4. Analysis of Code Contributor Authorship and Development Dynamics of Research Teams

4.1. Software Development Dynamics Within Research Teams

Understanding the composition and dynamics of software development teams provides essential context for analyzing how code contributions relate to scientific recognition and impact. To ensure reliable analysis, we focus on a subset of our article-repository pairs that meet several filtering conditions. First, we require that each article-repository pair have at least one citation, which helps ensure the research has received a basic level of engagement from the scientific community. Next, we require that repository commit activity must stop prior to 90 days past the date of article publication. Disallowing long-term projects ensures we do not include projects that may add additional code contributors later while still allowing a grace period during which developers can update repositories with additional documentation and publication information. We then subset the data to only include article-repository pairs with research teams of typical size by removing those with fewer than three authors and more than 12 authors, the 97th percentile for research team size. Finally, we filter out any author-developer pairs associated with these projects with predictive model confidence of less than 0.97 to ensure that we only include high-confidence matches[1]. This filtering process results in a dataset of 22804 article-repository pairs. A table with the counts of article-repository pairs, authors, and developers by data sources, domains, document types, and access status for this filtered dataset is shown in Table 5.

Within this filtered dataset, we categorized individuals into three groups: code-contributing authors (CC-A) who both authored papers and contributed code to associated repositories, non-code-contributing authors

[1]Figure 3 shows the distribution of predictive model confidence scores for author-developer pairs to justify this threshold. We chose the 0.97 threshold to ensure that we only include high-confidence matches while retaining a large proportion of the data (~90,000 author-developer pairs) as less than 3000 author-developer-account pairs have a confidence less than 0.97 in the whole unfiltered dataset.

(NCC-A) who authored papers but showed no evidence of code contributions, and code-contributing non-authors (CC-NA) who contributed code but received no authorship recognition. This categorization revealed that papers in our dataset typically have 4.9 ± 1.9 total authors, with 1.0 ± 0.7 code-contributing authors and 3.9 ± 2.0 non-code-contributing authors. Beyond the author list, papers averaged 0.4 ± 1.7 code-contributing non-authors. Table 2 details these distributions by domain, article type, and open access status.

Perhaps most striking is our finding that 6529 papers (28.6%) have at least one code contributor who was not matched to any article author. Within this substantial subset of papers, we found an average of 1.6 ± 2.8 un-matched code contributors per paper. To better understand the nature of these contributions, we conducted an analysis using a random sample of 200 code-contributing non-authors (see Section 7.4 for full methodology).

Our analysis revealed three distinct groups: 39% represent true non-authors (individuals who likely shouldn't be matched to an author), 30.5% appear to be missed classifications (individuals who likely should have been matched to authors), and the remaining 30.5% were unclear due to limited profile information. The majority of contributors across all groups made code changes rather than just documentation updates. Among true non-authors, 77.6% contributed code, with the top quartile contributing approximately 10% of total repository commits and 14% of absolute code changes. The unclear cases showed substantially higher contribution levels— many were repository owners with extensive engagement, yet even among the non-owners the median contributor accounted for approximately 35% of repository commits and 13% of absolute changes. Due to our limited sample size, we do not generalize these findings to the entire population but note that they describe the diverse nature of code contribution patterns from potentially unacknowledged contributors.

These findings reveal a more complex dynamic between software development and authorship recognition than previously documented. While our finding that each paper averages only a single code-contributing author aligns with previous research showing technical tasks typically fall to first authors (Larivière et al., 2020), the substantial contributions made by unrecognized contributors suggests systematic gaps in how scientific software development work is credited.

Table 2: Mean and Standard Deviation of Non-Code-Contributing Authors (NCC-A), Code-Contributing Authors (CC-A), and Code-Contributing Non-Authors (CC-NA) Research Team Members by Domain, Article Type, and Open Access Status. Only includes research teams from article-repository pairs with a most recent commit no later than 90 days after publication and excludes research teams in the top 3% of total author sizes.

Control	Subset	Total Authors	NCC-A	CC-A	CC-NA
OA Status	Closed	5.1 ± 1.9	4.1 ± 1.9	1.0 ± 0.7	0.5 ± 2.0
	Open	4.9 ± 1.9	3.9 ± 2.0	1.0 ± 0.7	0.4 ± 1.6
Domain	Health Sciences	6.1 ± 2.5	5.1 ± 2.6	0.9 ± 0.6	0.4 ± 1.2
	Life Sciences	5.2 ± 2.1	4.2 ± 2.2	1.0 ± 0.7	0.4 ± 1.2
	Physical Sciences	4.8 ± 1.8	3.8 ± 1.9	1.0 ± 0.7	0.5 ± 1.8
	Social Sciences	4.5 ± 1.7	3.5 ± 1.8	1.1 ± 0.7	0.3 ± 1.1
Article Type	preprint	4.8 ± 1.8	3.8 ± 1.9	1.0 ± 0.7	0.6 ± 2.1
	research article	4.9 ± 1.9	3.9 ± 2.0	1.0 ± 0.7	0.4 ± 1.6
	software article	4.7 ± 1.9	3.2 ± 1.9	1.5 ± 1.4	0.9 ± 1.1
Overall		**4.9 ± 1.9**	**3.9 ± 2.0**	**1.0 ± 0.7**	**0.4 ± 1.7**

When examining these patterns over time and across different team sizes (Figure 1), we found that the number of code-contributing authors and unrecognized contributors has remained relatively stable. This stability over time suggests that while the exclusion of code contributors from authorship is not worsening, it represents a persistent feature of scientific software development rather than a historical artifact or transition period in research practices. Similarly, the number of code-contributing non-authors remains constant even as team size grows, indicating that larger research teams do not necessarily adopt more inclusive authorship practices for code contributors, despite representing broader collaborative efforts.

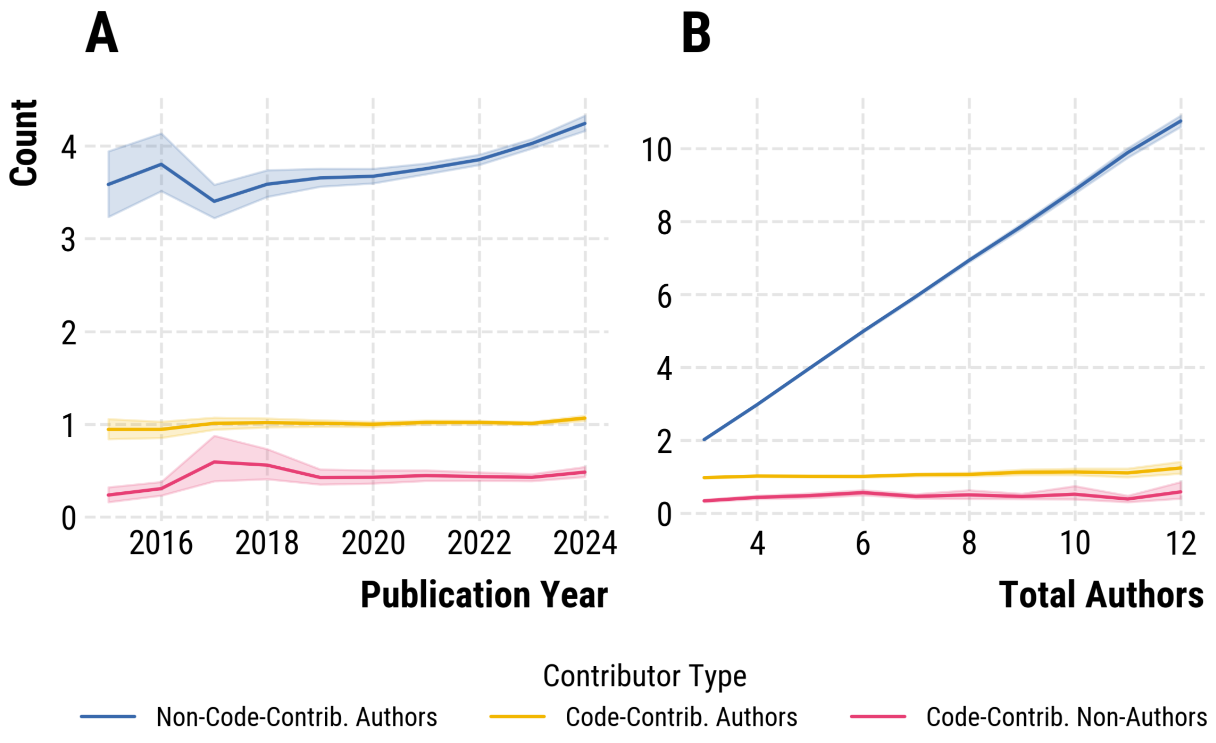

Figure 1: Average number of contributors per article, by contribution type along with A) the year the article was published, and B) the total number of authors included on the article. Only includes research teams from article-repository pairs with a most recent commit no later than 90 days after publication and excludes research teams in the top 3% of total author sizes for publication years with 50 or more articles. Shaded areas show the 95% confidence interval for the mean.

4.1.1. Modeling Article Citations

Building upon previous work examining the effects of team size and team diversity on scientific impact and software quality (see Section 2), we investigate how the number of code contributors within a research team may be associated with an article's research impact. We hypothesized that more code contributors might signal greater technical complexity in research, which may be associated with higher citation counts as the community builds upon more technically sophisticated works.

Using our filtered dataset of article-repository pairs (Table 5), we conducted multiple regression analyses to examine these relationships while controlling for various factors. Without controlling for domain, open access, or article type differences (Table 9), our analysis revealed a modest positive association between the number of code contributing authors and article citations, with each code-contributing author associated with a 5.1% increase in article citations ($p < 0.001$).

When controlling for article type (Table 12), we observed divergent patterns between preprints and research articles. For preprints, each code-contributing non-author was associated with a statistically significant 3.2% decrease in citations ($p < 0.005$). In contrast, research articles showed more positive associations: we found a significant positive relationship between code-contributing authors and citations ($p < 0.001$), though we cannot estimate the precise magnitude due to the non-significant main effect in the model. Additionally, each code-contributing non-author was associated with a 0.1% increase in expected citations for research articles ($p < 0.001$).

Based on these findings, we ***partially accept*** our hypothesis (*H1*) that "the number of individuals contributing code to a publication's associated repository positively correlates with the article's citation count." Several important nuances qualify this acceptance: the relationship is statistically significant but modest in magnitude and differs substantially between research articles (positive association) and preprints (negative association for non-author code contributors). These variations suggest that the relationship between code contributions and citation impact is context-dependent and more complex than initially hypothesized.

4.2. Characteristics of Scientific Code Contributors

4.2.1. Author Positions of Code Contributing Authors

Building upon previous work examining the relationship between authorship position and research contributions, we investigate how author position may relate to code contribution patterns. We hypothesized that first authors, traditionally contributing the bulk of intellectual and experimental work, are most likely to contribute code to a project. In contrast, middle and last authors often provide oversight and guidance and would be less likely to contribute code.

To analyze these patterns within our previously filtered dataset of article-repository pairs (Table 5), we conducted Chi-square tests of independence between author position and code contribution status. These tests revealed significant associations between author position and likelihood of code contribution overall and when controlling for research domain, article type, and open access status (all $p < 0.01$), indicating that the proportion of authors contributing code differs significantly based on author position. Following these significant associations, we examined the specific proportions across positions (Table 13): 68.6% of first authors contributed code to their projects, compared to only 9.3% of middle authors and 7.6% of last authors. The differences in these proportions remained statistically significant across all tested scenarios, regardless of research domain, article type, or open access status.

Based on these findings, we ***accept*** our hypothesis (*H2a*) that "first authors have higher code contribution rates than authors in other positions." The data demonstrates that the proportion of first authors who contribute code (68.6%) is significantly higher than the proportion of both middle authors (9.3%) and last authors (7.6%). This relationship remains robust and statistically significant across all tested conditions, including variations in research domain, article type, and open access status, indicating a fundamental connection between authorship position and technical contribution in scientific research.

4.2.2. Corresponding Status of Code Contributing Authors

Building upon our analysis of author position, we next examine how corresponding author status relates to code contribution patterns. We hypothesized that corresponding authors, who traditionally maintain research artifacts and serve as primary points of contact, would be more likely to contribute code compared to non-corresponding authors, as this role often involves responsibility for project resources and materials.

To analyze these relationships within our filtered dataset of article-repository pairs, we conducted Chi-square tests of independence between corresponding author status and code contribution status. Our analysis revealed patterns contrary to our initial hypothesis. The proportion of code contributors was low among both groups, with only 27.2% of corresponding authors and 20.0% of non-corresponding authors contributing code to their projects. Further examination (Table 14) showed that this pattern holds across nearly all conditions, with only a single exception: corresponding authors in closed-access publications showed no significant difference in their proportion of code contributors. However, this was tested with a sample of less than 200 authors.

Based on these findings, we ***reject*** our hypothesis (*H2b*) that "corresponding authors have higher code contribution rates than non-corresponding authors." Contrary to our expectations, our analysis revealed that the proportion of code contributors among corresponding authors (27.2%) did not significantly differ from the proportion among non-corresponding authors (20.0%). This pattern of similar proportions remained consistent across most studied conditions, with a single, small sample size exception in closed-access publications.

4.2.3. Modeling Author H-Index

Building upon previous work examining career implications for researchers who prioritize software development (see Section 2), we investigated how varying levels of code contribution relate to scholarly impact through h-index metrics. To ensure a robust analysis, we applied several key data filtering steps. We only included researchers with at least three publications in our dataset, removed those with more than three developer account associations, and used each researcher's most common domain, article type, and author position, with ties broken by the most recent occurrence. We removed h-index outliers by excluding researchers below the bottom 3rd and above the top 97th percentiles. Finally, we removed any author-developer-account pairs with a predictive model confidence of less than 0.97. Table 15 summarizes the number of researchers in each coding frequency group, categorized by author position, publication type, and research domain.

We categorized researchers' coding contributions into mutually exclusive groups: non-coders (no code contributions), any coding (code contribution in less than half of article-repository pairs), majority coding (code contribution in at least half, but not all, article-repository pairs), and always coding (code contribution in every article-repository pair).

Figure 2 shows the distribution of author h-indices across these coding frequency groups, grouped by author position, publication type, and research domain.

Our analysis revealed a consistent and statistically significant negative relationship between code contribution frequency and h-index across multiple analytical controls. Our initial uncontrolled analysis (Table 16) indicates increasingly adverse h-index effects as researcher coding frequency increases. Compared to non-coding authors, researchers were associated with progressively lower h-indices: occasional code contributors showed a ~27.3% lower h-index ($p < 0.001$), majority code contributors demonstrated a ~53.5% lower h-index ($p < 0.001$), and always coding authors exhibited a ~62.1% lower h-index ($p < 0.001$).

When controlling for author position (Table 17), we found a general pattern of reduced h-indices with increased code contribution, with one notable exception. Occasional coding first authors were associated with a ~14.9% higher h-index ($p < 0.001$), while always coding first authors saw a ~21.6% reduction compared to non-coding first authors ($p < 0.001$). For middle and last authors, the pattern was more consistently negative. Middle authors who occasionally coded showed a ~26.6% lower h-index ($p < 0.001$), and those always coding demonstrated a ~52.9% lower h-index ($p < 0.001$). Similarly, last authors who occasionally coded experienced a ~13.1% lower h-index ($p < 0.001$), with always coding authors showing a ~45.7% lower h-index ($p < 0.001$).

When controlling for research domain (Table 18), majority coding scientists showed significant h-index reductions across all domains. Health sciences researchers saw the most dramatic reduction at ~76.5% ($p < 0.001$), followed by physical sciences at ~52.6% ($p < 0.001$), social sciences at ~51.4% ($p < 0.001$), and life sciences at ~47.1% ($p < 0.001$).

Analyzing by common article type (Table 19) revealed similar patterns. For authors primarily publishing preprints, the h-index reductions were substantial: ~25.6% for occasional coding, ~53.5% for majority coding, and ~62.9% for always coding authors. Authors primarily publishing software articles showed slightly better but still significant reductions: ~33.1% for majority coding and ~33.0% for always coding authors.

Based on these findings, we ***accept*** our hypothesis (*H3*) that "the frequency with which individual researchers contribute code to their research projects is negatively correlated with their h-index." Our analysis demonstrates a clear and statistically significant negative relationship between coding frequency and scholarly impact as measured by the researcher's h-index. This relationship was robust across multiple analytical controls, including author position, research domain, and article type. These results are particularly striking because each of our models includes publication count as an input feature, suggesting that these h-index reductions persist even when accounting for total research output.

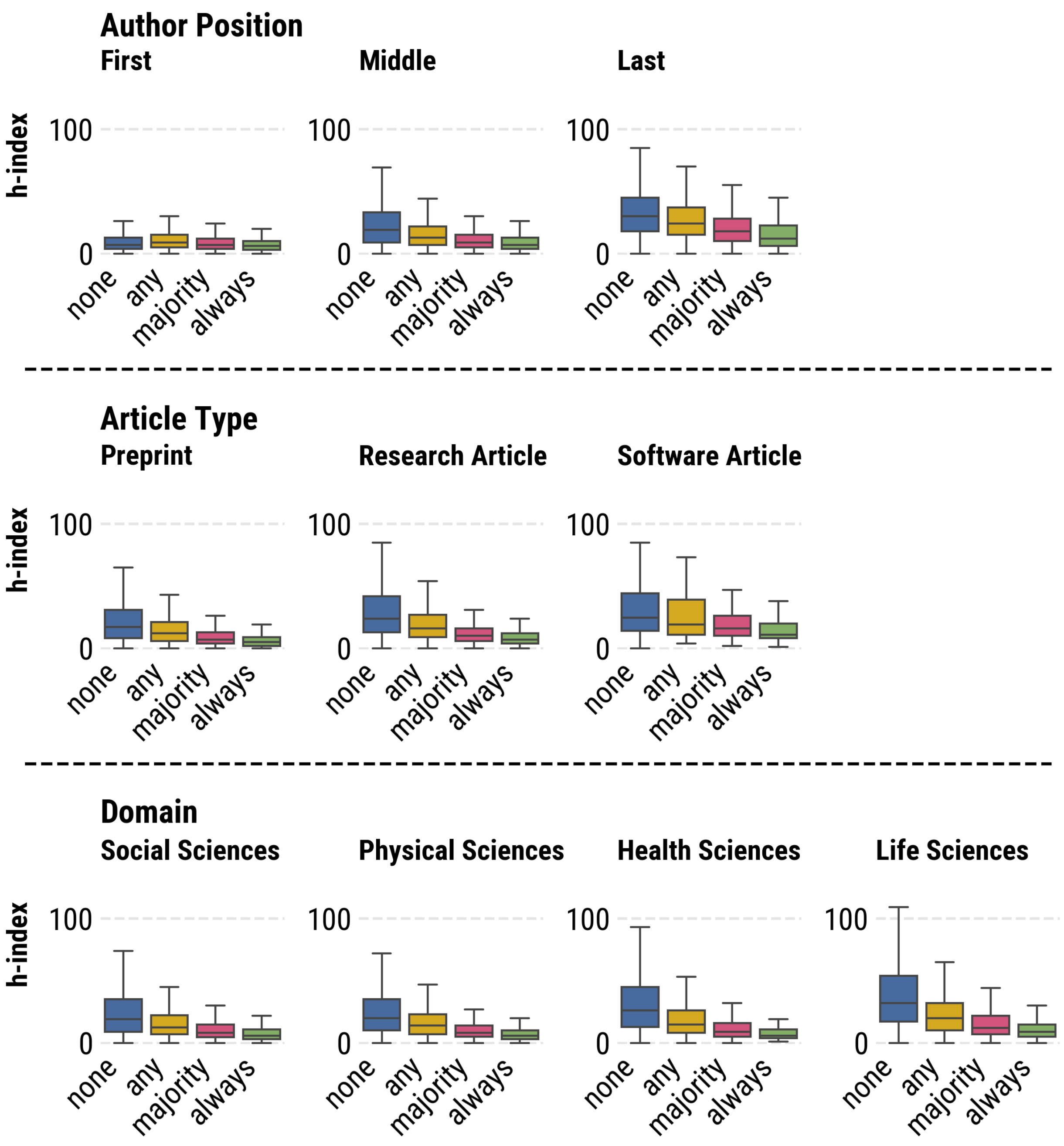

Figure 2: Distribution of author h-index by coding frequency across three key publication factors. Results are grouped by each author's most frequent: (1) position in publication bylines (first, middle, or last), (2) publication type (preprint, research article, or software article), and (3) research domain (Social Sciences, Physical Sciences, Health Sciences, or Life Sciences). Within each subplot, h-indices are divided by the author's coding frequency: 'none' (no coding in any of their publications), 'any' (coding in at least one but fewer than half of their publications), 'majority' (coding in at least half but not all of their publications), and 'always' (coding in each of their publications). Authors are only included if they have three or more publications within our dataset and are associated with no more than three developer accounts, with each association having a predicted model confidence of at least 97%.

5. Discussion

Our analysis reveals significant disparities in the recognition of software contributions to scientific research, with ~30% (n=6529) of articles having code-contributors not matched to any article author, partially representing unrecognized code contribution. This persistent pattern over time and team size suggests a systemic disconnect between software development and scientific recognition systems, reflecting challenges in how scientific contributions are valued and credited. This exclusion reflects what Shapin (1989) observed about scientific authority—the selective attribution of technical work as either genuine knowledge or mere skill significantly impacts who receives formal recognition. These findings further support previous research by Philippe et al. (2019) and Carver et al. (2022) documenting the frequent relegation of software contributors to either acknowledgment sections or receiving no credit at all, rather than authorship positions, despite the increasingly central role of software in scientific inquiry. The stability of this pattern over time indicates that this phenomenon has embedded itself in scientific software development rather than representing a transitional phase, raising questions about scientific labor and how reward structures integrate technical contributions.

Our finding that, on average, article-repository pairs have only a single code contributor mirrors prior work from Färber (2020). Further, the distribution of code contributions across author positions provides context to the hierarchical organization of scientific work. First authors emerge as significantly more likely to contribute code with 68.6% of all first authors in our dataset contributing code. Middle and last authors, meanwhile, were statistically significantly less likely to contribute code, with only 9.3% of middle authors and 7.6% of last authors acting as code-contributing members of the research team. Corresponding authors were similarly less likely than expected to be code contributors, as we found that within our dataset, corresponding authors were code contributors 27.2% of the time. These patterns align with traditional scientific labor distribution where first authors typically handle technical aspects of research while middle and last authors are likely specialist contributors or provide guidance and oversight (Larivière et al., 2020; Sauermann and Haeussler, 2017). However, our data did not support our initial hypothesis that corresponding authors would also be more likely to contribute code due to their shared responsibility for the long-term maintenance of research artifacts. This finding suggests a potential strict division between project management responsibilities and direct technical engagement with software development.

The modest citation advantage associated with code-contributing authors (5.1% increase in citations per code-contributing author) stands in contrast with the significant negative relationship between coding frequency and an individual's scholarly impact (h-index). This misalignment between technical contributions and scientific recognition creates an asymmetrical relationship in which software development may enhance research impact but potentially penalizes individual careers. The progressive reduction in h-index as coding frequency increases indicates a cumulative disadvantage for frequent code contributors. This pattern persists even when controlling for publication count, suggesting issues in how software contributions are valued relative to other scientific outputs. These findings echo concerns raised by Muna et al. (2016) about the sustainability of research software development and highlight how current reward structures may discourage talented developers from pursuing scientific careers.

Software development represents a form of scholarly labor that has become increasingly essential to modern research yet remains incompletely integrated into formal recognition systems. Similar to the high proportion of articles with authors who made data curation contributions towards research observed by Larivière et al. (2020), our finding that a quarter of papers have unacknowledged code contributors highlights a labor role that is simultaneously common and undervalued. The prevalence of code contributions across domains demonstrates the importance of this work to contemporary research. However, the persistent exclusion of contributors from authorship suggests that researchers continue to classify software development as technical support rather than intellectual contribution. This classification may reflect disciplinary traditions that privilege certain forms of scholarly production despite the growing recognition that software represents a legitimate research output (Katz et al., 2020). The tension between software's importance and contributors' recognition status raises questions about how we define, value, and reward different forms of scientific labor in an increasingly computational research landscape.

5.1. Limitations

Our data collection approach introduces several methodological constraints that should be considered when interpreting these results. By focusing exclusively on GitHub repositories, we likely miss contributions stored on alternative platforms such as GitLab, Bitbucket, or institutional repositories, potentially skewing our understanding of contribution patterns. As Trujillo et al. (2022), Cao et al. (2023), and Escamilla et al. (2022) have all noted, while GitHub is the predominate host of scientific software, significant portions of research code exist on other platforms. Additionally, our reliance on public repositories means we cannot account for private repositories or code that were never publicly shared, potentially underrepresenting sensitive research areas or proprietary methods.

Our predictive modeling approach for matching authors with developer accounts presents additional limitations. The model's performance can be affected by shorter names where less textual information is available for matching, potentially creating biases against researchers from cultures with shorter naming conventions. Organization accounts used for project management pose particular challenges for accurate matching, and while we implemented filtering mechanisms to minimize their impact, some misclassifications may persist. Furthermore, our approach may not capture all code contributors if multiple individuals developed code. However, only one uploaded it to a repository, creating attribution artifacts that may systematically underrepresent specific contributors, particularly junior researchers or technical staff who may not have direct repository access.

Our analytical approach required substantial data filtering to ensure reliable results, introducing potential selection biases in our sample. By focusing on article-repository pairs with commit activity no later than 90 days past the date of article publication and at least three authors and less than 12 authors, we may have systematically excluded certain types of research projects, particularly those with extended development timelines or extensive collaborations. Our categorization of coding status (non-coder, any coding, majority coding, always coding) necessarily simplifies complex contribution patterns. It does not account for code contributions' quality, complexity, or significance. Additionally, our reliance on OpenAlex metadata introduces certain limitations to our analysis. While OpenAlex provides good overall coverage, it lags behind proprietary databases in indexing references and citations. The lag in OpenAlex data may affect our citation-based analyses and the completeness of author metadata used in our study (Alperin et al., 2024).

5.2. Future Work

Future technical improvements may enhance our understanding of the relationship between software development and scientific recognition systems. Expanding analysis beyond GitHub to include other code hosting platforms would provide a more comprehensive understanding of scientific software development practices across domains and institutional contexts. More sophisticated entity-matching techniques could improve author-developer account identification, particularly for cases with limited information or common names. Developing more nuanced measures and classifications of code contribution type, quality, and significance beyond binary contribution identification would better capture the true impact of technical contributions to research (as we have started to do in Section 7.1.2.2). These methodological advances would enable more precise tracking of how code contributions translate—or fail to translate—into formal scientific recognition, providing clearer evidence for policy interventions.

Our findings point to several directions for future research on the changing nature of scientific labor and recognition. Longitudinal studies tracking how code contribution patterns affect career trajectories would provide valuable insights into the long-term impacts of the observed h-index disparities and whether these effects vary across career stages. Comparative analyses across different scientific domains could reveal discipline-specific norms and practices around software recognition, potentially identifying models that more equitably credit technical contributions. Qualitative studies examining how research teams make authorship decisions regarding code contributors would complement our quantitative findings by illuminating the social and organizational factors influencing recognition practices. Additionally, to better understand corresponding authors' role in maintaining research artifacts, future work could remove the 90-day post-publication commit activity filter to examine long-term sustainability actions. However, this approach would need to address the introduction of contributors unrelated to the original paper.

Despite their growing importance, the persistent underrecognition of software contributions suggests a need for structural interventions in how we conceptualize and reward scientific work. Building upon efforts like CRediT (Brand et al., 2015), future work should investigate potential policy changes to better align institutional incentives with the diverse spectrum of contributions that drive modern scientific progress. However, as the example of CRediT demonstrates, even well-intentioned taxonomies may reproduce existing hierarchies or create new forms of inequality if they fail to address underlying power dynamics in scientific communities. The challenge is not merely technical but social: creating recognition systems that simultaneously support innovation, ensure appropriate credit, maintain research integrity, and foster equitable participation in an increasingly computational scientific enterprise.

6. References

Ahuja, G., 2000. Collaboration networks, structural holes, and innovation: A longitudinal study. Administrative science quarterly 45, 425–455.

Aksnes, D.W., 2006. Citation rates and perceptions of scientific contribution. J. Assoc. Inf. Sci. Technol. 57, 169–185.

Allen, L., Scott, J., Brand, A., Hlava, M., Altman, M., 2014. Publishing: Credit where credit is due. Nature 508, 312–313.

Alperin, J.P., Portenoy, J., Demes, K., Larivière, V., Haustein, S., 2024. An analysis of the suitability of openalex for bibliometric analyses. arXiv preprint arXiv:2404.17663 .

Brand, A., Allen, L., Altman, M., Hlava, M., Scott, J., 2015. Beyond authorship: Attribution, contribution, collaboration, and credit. Learned Publishing 28.

Cannon, K., Caudill, S., Chan, C., Cousins, B., Creighton, J.D., Ewing, B., Fong, H., Godwin, P., Hanna, C., Hooper, S., et al., 2021. Gstlal: A software framework for gravitational wave discovery. SoftwareX 14, 100680.

Cao, H., Dodge, J., Lo, K., McFarland, D.A., Wang, L.L., 2023. The rise of open science: Tracking the evolution and perceived value of data and methods link-sharing practices. ArXiv abs/2310.03193.

Carver, J.C., Weber, N., Ram, K., Gesing, S., Katz, D.S., 2022. A survey of the state of the practice for research software in the united states. PeerJ Computer Science 8.

Chen, L., lan Ding, J., Song, D., Qu, Z., 2024. Exploring scientific contributions through citation context and division of labor. ArXiv abs/2410.13133.

Chinchilla-Rodríguez, Z., Costas, R., Lariviére, V., Robinson-García, N., Sugimoto, C.R., 2022. The relationship between corresponding authorship and author position, in: Proceedings of the 26th International Conference on Science, Technology and Innovation Indicators (STI 2022), pp. 7–9.

Devlin, J., Chang, M., Lee, K., Toutanova, K., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805. URL: http://arxiv.org/abs/1810.04805, arXiv:1810.04805.

Du, C., Cohoon, J., Lopez, P., Howison, J., 2021. Softcite dataset: A dataset of software mentions in biomedical and economic research publications. Journal of the Association for Information Science and Technology 72, 870–884.

Du, C., Cohoon, J., Lopez, P., Howison, J., 2022. Understanding progress in software citation: a study of software citation in the cord-19 corpus. PeerJ Computer Science 8, e1022.

Escamilla, E., Klein, M., Cooper, T., Rampin, V., Weigle, M.C., Nelson, M.L., 2022. The rise of github in scholarly publications, in: Silvello, G., Corcho, O., Manghi, P., Di Nunzio, G.M., Golub, K., Ferro, N., Poggi, A. (Eds.), Linking Theory and Practice of Digital Libraries, Springer International Publishing, Cham. pp. 187–200.

Färber, M., 2020. Analyzing the github repositories of research papers, in: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, Association for Computing Machinery, New York, NY, USA. p. 491–492. URL: https://doi.org/10.1145/3383583.3398578, doi:10.1145/3383583.3398578.

Franceschet, M., Costantini, A., 2010. The effect of scholar collaboration on impact and quality of academic papers. J. Informetrics 4, 540–553.

Hampton, S.E., Strasser, C.A., Tewksbury, J.J., Gram, W.K., Budden, A.E., Batcheller, A.L., Duke, C.S., Porter, J.H., 2013. Big data and the future of ecology. Frontiers in Ecology and the Environment 11, 156–162.

Hannay, J.E., MacLeod, C., Singer, J., Langtangen, H.P., Pfahl, D., Wilson, G., 2009. How do scientists develop and use scientific software?, in: 2009 ICSE workshop on software engineering for computational science and engineering, Ieee. pp. 1–8.

Hasselbring, W., Druskat, S., Bernoth, J., Betker, P., Felderer, M., Ferenz, S., Lamprecht, A.L., Linxweiler, J., Rumpe, B., 2024. Toward research software categories. URL: https://arxiv.org/abs/2404.14364, arXiv:2404.14364.

Hata, H., Guo, J.L.C., Kula, R.G., Treude, C., 2021. Science-software linkage: The challenges of traceability between scientific knowledge and software artifacts. ArXiv abs/2104.05891.

He, P., Gao, J., Chen, W., 2021a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. arXiv:2111.09543.

He, P., Liu, X., Gao, J., Chen, W., 2021b. Deberta: Decoding-enhanced bert with disentangled attention, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=XPZIaotutsD.

Herbsleb, J.D., Mockus, A., 2003. An empirical study of speed and communication in globally distributed software development. IEEE Transactions on software engineering 29, 481–494.

Hettrick, S., et al., 2014. It's impossible to conduct research without software, say 7 out of 10 uk researchers. Software Sustainability Institute. Retrieved October 20, 2016.

Howison, J., Herbsleb, J.D., 2011. Scientific software production: incentives and collaboration, in: Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, Association for Computing Machinery, New York, NY, USA. p. 513–522. URL: https://doi.org/10.1145/1958824.1958904, doi:10.1145/1958824.1958904.

Howison, J., Herbsleb, J.D., 2013. Incentives and integration in scientific software production. Proceedings of the 2013 conference on Computer supported cooperative work .

Jin, X., Wah, B.W., Cheng, X., Wang, Y., 2015. Significance and challenges of big data research. Big data research 2, 59–64.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al., 2021. Highly accurate protein structure prediction with alphafold. nature 596, 583–589.

Júnior, E.A.C., Silva, F.N., da Fontoura Costa, L., Amancio, D.R., 2016. Patterns of authors contribution in scientific manuscripts. J. Informetrics 11, 498–510.

Katz, D.S., Hong, N.P.C., Clark, T., Muench, A., Stall, S., Bouquin, D.R., Cannon, M., Edmunds, S.C., Faez, T., Feeney, P., Fenner, M., Friedman, M., Grenier, G., Harrison, M., Heber, J., Leary, A., MacCallum, C.J., Murray, H., Pastrana, É., Perry, K., Schuster, D.C., Stockhause, M., Yeston, J.S., 2020. Recognizing the value of software: a software citation guide. F1000Research 9.

Kelley, A., Garijo, D., 2021. A framework for creating knowledge graphs of scientific software metadata. Quantitative Science Studies 2, 1423–1446.

Lamprecht, A.L., García, L.J., Kuzak, M., Martinez, C., Arcila, R., Pico, E.M.D., Angel, V.D.D., van de Sandt, S., Ison, J.C., Martínez, P.A., McQuilton, P., Valencia, A., Harrow, J.L., Psomopoulos, F., Gelpi, J.L., Hong, N.P.C., Goble, C.A., Capella-Gutiérrez, S., 2020. Towards fair principles for research software. Data Sci. 3, 37–59.

Larivière, V., Gingras, Y., Sugimoto, C.R., Tsou, A., 2014. Team size matters: Collaboration and scientific impact since 1900. Journal of the Association for Information Science and Technology 66.

Larivière, V., Pontille, D., Sugimoto, C.R., 2020. Investigating the division of scientific labor using the contributor roles taxonomy (credit). Quantitative Science Studies , 1–18.

Larivière, V., Desrochers, N., Macaluso, B., Mongeon, P., Paul-Hus, A., Sugimoto, C.R., 2016. Contributorship and division of labor in knowledge production. Social Studies of Science 46, 417–435. URL: https://doi.org/10.1177/0306312716650046, doi:10.1177/0306312716650046, arXiv:https://doi.org/10.1177/0306312716650046. pMID: 28948891.

Liu, X., Zhang, C., Li, J., 2023. Conceptual and technical work: Who will disrupt science? Journal of Informetrics 17, 101432. URL: https://www.sciencedirect.com/science/article/pii/S1751157723000573, doi:10.1016/j.joi.2023.101432.

Merton, R.K., 1968. The matthew effect in science: The reward and communication systems of science are considered. Science 159, 56–63.

Milewicz, R., Pinto, G., Rodeghero, P., 2019. Characterizing the roles of contributors in open-source scientific software projects, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pp. 421–432. doi:10.1109/MSR.2019.00069.

Milewicz, R., Raybourn, E., 2018. Talk to me: A case study on coordinating expertise in large-scale scientific software projects, in: 2018 IEEE 14th International Conference on e-Science (e-Science), IEEE. pp. 9–18.

Momcheva, I., Tollerud, E., 2015. Software use in astronomy: an informal survey. arXiv preprint arXiv:1507.03989 .

Muna, D., Alexander, M., Allen, A., Ashley, R., Asmus, D., Azzollini, R., Bannister, M., Beaton, R., Benson, A., Berriman, G.B., Bilicki, M., Boyce, P., Bridge, J., Cami, J., Cangi, E., Chen, X., Christiny, N., Clark, C., Collins, M., Comparat, J., Cook, N., Croton, D., Davids, I.D., Éric Depagne, Donor, J., dos Santos, L.A., Douglas, S., Du, A., Durbin, M., Erb, D., Faes, D., Fernández-Trincado, J.G., Foley, A., Fotopoulou, S., Frimann, S., Frinchaboy, P., Garcia-Dias, R., Gawryszczak, A., George, E., Gonzalez, S., Gordon, K., Gorgone, N., Gosmeyer, C., Grasha, K., Greenfield, P., Grellmann, R., Guillochon, J., Gurwell, M., Haas, M., Hagen, A., Haggard, D., Haines, T., Hall, P., Hellwing, W., Herenz, E.C., Hinton, S., Hlozek, R., Hoffman, J., Holman, D., Holwerda, B.W., Horton, A., Hummels, C., Jacobs, D., Jensen, J.J., Jones, D., Karick, A., Kelley, L., Kenworthy, M., Kitchener, B., Klaes, D., Kohn, S., Konorski, P., Krawczyk, C., Kuehn, K., Kuutma, T., Lam, M.T., Lane, R., Liske, J., Lopez-Camara, D., Mack, K., Mangham, S., Mao, Q., Marsh, D.J.E., Mateu, C., Maurin, L., McCormac, J., Momcheva, I., Monteiro, H., Mueller, M., Munoz, R., Naidu, R., Nelson, N., Nitschelm, C., North, C., Nunez-Iglesias, J., Ogaz, S., Owen, R., Parejko, J., Patrício, V., Pepper, J., Perrin, M., Pickering, T., Piscionere, J., Pogge, R., Poleski, R., Pourtsidou, A., Price-Whelan, A.M., Rawls, M.L., Read, S., Rees, G., Rein, H., Rice, T., Riemer-Sørensen, S., Rusomarov, N., Sanchez, S.F., Santander-García, M., Sarid, G., Schoenell, W., Scholz, A., Schuhmann, R.L., Schuster, W., Scicluna, P., Seidel, M., Shao, L., Sharma, P., Shulevski, A., Shupe, D., Sifón, C., Simmons, B., Sinha, M., Skillen, I., Soergel, B., Spriggs, T., Srinivasan, S., Stevens, A., Streicher, O., Suchyta, E., Tan, J., Telford, O.G., Thomas, R., Tonini, C., Tremblay, G., Tuttle, S., Urrutia, T., Vaughan, S., Verdugo, M., Wagner, A., Walawender, J., Wetzel, A., Willett, K., Williams, P.K.G., Yang, G., Zhu, G., Zonca, A., 2016. The astropy problem. URL: https://arxiv.org/abs/1610.03159, arXiv:1610.03159.

Neang, A.B., Sutherland, W., Ribes, D., Lee, C.P., 2023. Organizing oceanographic infrastructure: the work of making a software pipeline repurposable. Proceedings of the ACM on Human-Computer Interaction 7, 1–18.

Newman, D., Herrera, C.N., Parente, S.T., 2014. Overcoming barriers to a research-ready national commercial claims database. American Journal of Managed Care 20, ESP25–ESP30. WOS:000351003100004.

Paine, D., Lee, C.P., 2017. " who has plots?" contextualizing scientific software, practice, and visualizations. Proceedings of the ACM on human-computer interaction 1, 1–21.

Park, H., Wolfram, D., 2019. Research software citation in the data citation index: Current practices and implications for research software sharing and reuse. Journal of Informetrics 13, 574–582. URL: https://www.sciencedirect.com/science/

article/pii/S1751157718302372, doi:10.1016/j.joi.2019.03.005.

Philippe, O., Hammitzsch, M., Janosch, S., van der Walt, A., van Werkhoven, B., Hettrick, S., Katz, D.S., Leinweber, K., Gesing, S., Druskat, S., Henwood, S., May, N.R., Lohani, N.P., Sinha, M., 2019. softwaresaved/international-survey: Public release for 2018 results. URL: https://doi.org/10.5281/zenodo.2585783, doi:10.5281/zenodo.2585783.

Reimers, N., Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics. URL: https://arxiv.org/abs/1908.10084.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv abs/1910.01108.

Sauermann, H., Haeussler, C., 2017. Authorship and contribution disclosures. Science Advances 3, e1700404. URL: https://www.science.org/doi/abs/10.1126/sciadv.1700404, doi:10.1126/sciadv.1700404, arXiv:https://www.science.org/doi/pdf/10.1126/sciadv.1700404.

Shapin, S., 1989. The invisible technician. American scientist 77, 554–563.

Sharma, N.K., Ayyala, R., Deshpande, D., Patel, Y., Munteanu, V., Ciorba, D., Bostan, V., Fiscutean, A., Vahed, M., Sarkar, A., et al., 2024. Analytical code sharing practices in biomedical research. PeerJ Computer Science 10, e2066.

Shen, H.W., Barabási, A.L., 2014. Collective credit allocation in science. Proceedings of the National Academy of Sciences 111, 12325–12330.

Smith, A.M., Katz, D.S., Niemeyer, K.E., 2016. Software citation principles. PeerJ Comput. Sci. 2, e86.

Stankovski, A., Garijo, D., 2024. Repofrompaper: An approach to extract software code implementations from scientific publications, in: NSLP.

Stodden, V., Guo, P., Ma, Z., 2013. Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals. PloS one 8, e67111.

Trujillo, M.Z., Hébert-Dufresne, L., Bagrow, J., 2022. The penumbra of open source: projects outside of centralized platforms are longer maintained, more academic and more collaborative. EPJ Data Science 11, 31.

West, J.D., Jacquet, J., King, M.M., Correll, S.J., Bergstrom, C.T., 2013. The role of gender in scholarly authorship. PloS one 8, e66212.

Wyss, E., De Carli, L., Davidson, D., 2023. (nothing but) many eyes make all bugs shallow, in: Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses, Association for Computing Machinery, New York, NY, USA. p. 53–63. URL: https://doi.org/10.1145/3605770.3625216, doi:10.1145/3605770.3625216.

7. Appendix

7.1. Extended Data and Methods

7.1.1. Building a Dataset of Linked Scientific Articles and Code Repositories

The increasing emphasis on research transparency has led many journals and platforms to require or recommend code and data sharing (Stodden et al., 2013; Sharma et al., 2024), creating traceable links between publications and code. These explicit links enable systematic study of both article-repository and author-developer account relationships (Hata et al., 2021; Kelley and Garijo, 2021; Stankovski and Garijo, 2024; Milewicz et al., 2019).

Our dataset collection process leveraged four sources of linked scientific articles and code repositories, each with specific mechanisms for establishing these connections:

1. **Public Library of Science (PLOS)**: We extracted repository links from PLOS articles' mandatory data and code availability statements.
2. **Journal of Open Source Software (JOSS)**: JOSS requires explicit code repository submission and review as a core part of its publication process.
3. **SoftwareX**: Similar to JOSS, SoftwareX mandates code repositories as a publication requirement.
4. **Papers with Code**: This platform directly connects machine learning preprints with their implementations. We focus solely on the "official" article-repository relationships rather than the "unverified" or "unofficial" links.

To create a comprehensive and analyzable dataset, we enriched these article-repository pairs with metadata from multiple sources. We utilized the Semantic Scholar API for DOI resolution to ensure we found the latest version of each article. This resolution step was particularly important when working with preprints, as journals may have published these papers since their inclusion in the Papers with Code dataset. Using

Semantic Scholar, we successfully resolved 56.3% of all DOIs within our dataset[2].

We then utilized the OpenAlex API to gather detailed publication metadata, including:

- Publication characteristics (open access status, domain, publication date)
- Author details (name, author position, corresponding author status)
- Article- and individual-level metrics (citation counts, FWCI, h-index)

Similarly, the GitHub API provided comprehensive information for source code repositories:

- Repository metadata (name, description, programming languages, creation date)
- Contributor details (username, display name, email)
- Repository-level metrics (star count, fork count, issue count)

7.1.2. Developing a Predictive Model for Author-Developer Account Matching

7.1.2.1. Annotated Dataset Creation.

Creating an accurate author-developer account matching model required quality-labeled training data that reflects real-world identity matching challenges. Exact-matching on names or emails proved insufficient due to variations in formatting (e.g., "J. Doe" vs. "Jane Doe"), use of institutional versus personal email addresses, and incomplete information. However, author and developer account information often contains sufficient similarities for probabilistic matching, such as when author "Jane Doe" corresponds to username "jdoe" or "janedoe123".

To efficiently build our training and evaluation dataset, we used JOSS articles as we believed they typically feature higher author-developer-account overlap, increasing positive match density. Our dataset creation process followed these steps:

1. We generated semantic embeddings for each developer account and author name using the multi-qa-MiniLM-L6-cos-v1 model from the Sentence Transformers Python library (Reimers and Gurevych, 2019).
2. We calculated cosine similarity between all potential author-developer-account pairs for each article-repository pair.
3. For annotation efficiency, we selected the three most similar authors for each developer account.

From these generated author-developer-account pairs, we randomly selected 3,000 for classification by two independent annotators as either matches or non-matches, resolving disagreements through discussion and verification. The resulting dataset contains 451 (15.0%) positive matches and 2548 (85.0%) negative matches, comprising 2027 unique authors and 2733 unique developer accounts.

Our collected data for annotation confirmed that exact matching would be insufficient—only 2191 (80.2%) of developer accounts had associated display names and just 839 (30.7%) had associated email addresses.

7.1.2.2. Training and Evaluation.

Our training and evaluation methodology began with careful dataset preparation to prevent data leakage between training and test sets. To ensure complete separation of authors and developers, we randomly selected 10% of unique authors and 10% of unique developers, designating any pairs containing these selected entities for the test set. This entity-based splitting strategy resulted in 2442 (81.4%) pairs for training and 557 (18.6%) pairs for testing.

For our predictive model, we evaluated three transformer-based architectures that have demonstrated strong performance in entity matching tasks:

[2]Broken out by dataset source, we resolved 2.1% of all PLOS DOIs, 4.0% of all JOSS DOIs, 0.0% of all SoftwareX DOIs, and 49.2% of all Papers with Code (arXiv) DOIs.

- DeBERTa-v3-base (He et al., 2021a,b)
- mBERT (bert-base-multilingual-cased) (Devlin et al., 2018)
- DistilBERT (Sanh et al., 2019)

We systematically evaluated these base models across different combinations of developer-account features, ranging from using only the username to incorporating complete profile information (username, display name, and email address). We fine-tuned all models using the Adam optimizer with a linear learning rate of 1e-05 for training and a batch size of 8 for training and evaluation. Given the size of our dataset and the binary nature of our classification task, models were trained for a single epoch to prevent overfitting.

We evaluated model performance using standard binary classification metrics: precision, recall, and F1-score. This evaluation framework allowed us to directly compare model architectures and feature combinations while accounting for the balance between precision and recall in identifying correct matches.

Our comprehensive model evaluation revealed that fine-tuning DeBERTa-v3-base (He et al., 2021a) with developer username and display name as input features produces optimal performance for author-developer matching. This model configuration achieved a binary F1 score of 0.944, with an accuracy of 0.984, precision of 0.938, and recall of 0.95. Table 3 presents a complete comparison of model architectures and feature combinations.

Table 3: Comparison of Models for Author-Developer-Account Matching

Optional Feats.	**Model**	**Accuracy**	**Precision**	**Recall**	**F1**
name	deberta	0.984	0.938	0.950	0.944
name, email	bert-multilingual	0.984	0.938	0.950	0.944
name, email	deberta	0.982	0.907	0.975	0.940
name	bert-multilingual	0.982	0.938	0.938	0.938
name	distilbert	0.978	0.936	0.912	0.924
name, email	distilbert	0.978	0.936	0.912	0.924
email	deberta	0.957	0.859	0.838	0.848
email	bert-multilingual	0.950	0.894	0.738	0.808
n/a	deberta	0.946	0.847	0.762	0.803
n/a	bert-multilingual	0.941	0.862	0.700	0.772
n/a	distilbert	0.856	0.000	0.000	0.000
email	distilbert	0.856	0.000	0.000	0.000

Analysis of each model's performance revealed that including developer display names had the largest positive impact on model performance compared to username alone. We also observed that mBERT's performance was comparable to DeBERTa's while using the developer email address as an additional input feature. However, we selected the DeBERTa configuration as it had consistent strong performance across various feature combinations.

To facilitate the reuse of our work, we have made our trained model and supporting code publicly available. Complete fine-tuning, evaluation, and inference code is available as the Python package: sci-soft-models, and the fine-tuned model has been released on HuggingFace (evamxb/dev-author-em-clf).

7.1.2.3. Model Limitations.

While our model demonstrates strong performance, we acknowledge certain limitations in our approach:

1. **Short name sensitivity**: Shorter names (both usernames and display names) can affect the model's performance, as less textual information is available for matching.

2. **Organization accounts**: Research lab accounts used for project management present a potential challenge for accurate matching, as they don't correspond to individual authors. However, our filtering mechanisms applied before analysis help minimize their impact in modeling.

7.1.3. Dataset Characteristics and Repository Types

Our compiled dataset appears to contain a mix of repository types, varying from analysis script repositories to software tools and likely some "code dumps" (where code is copied to a new repository immediately before publication). This diversity is reflected in the commit duration patterns across different publication types. The median commit duration for repositories in our analysis is:

- 47 days for preprints
- 104 days for research articles
- 247 days for software articles

Complete statistics on commit durations, including count, mean, and quantile details, are available in Table 4.

Table 4: Commit duration (in days) distributions for different publication types. Only includes article-repository pairs with a most recent commit no later than 90 days after publication and excludes publications from research teams in the top 3% of total author sizes.

	count	mean	std	min	10%	25%	50%	75%	90%	max
article_type										
preprint	3080.0	104.860390	178.024291	-1520.0	0.0	5.0	47.0	130.00	270.0	2091.0
research article	19502.0	184.375551	247.760274	-931.0	0.0	13.0	104.0	258.75	474.0	3176.0
software article	222.0	372.509009	470.233626	-1.0	0.0	22.0	247.0	510.75	904.7	3007.0

7.2. Distributions of Author-Developer-Account Prediction Confidence

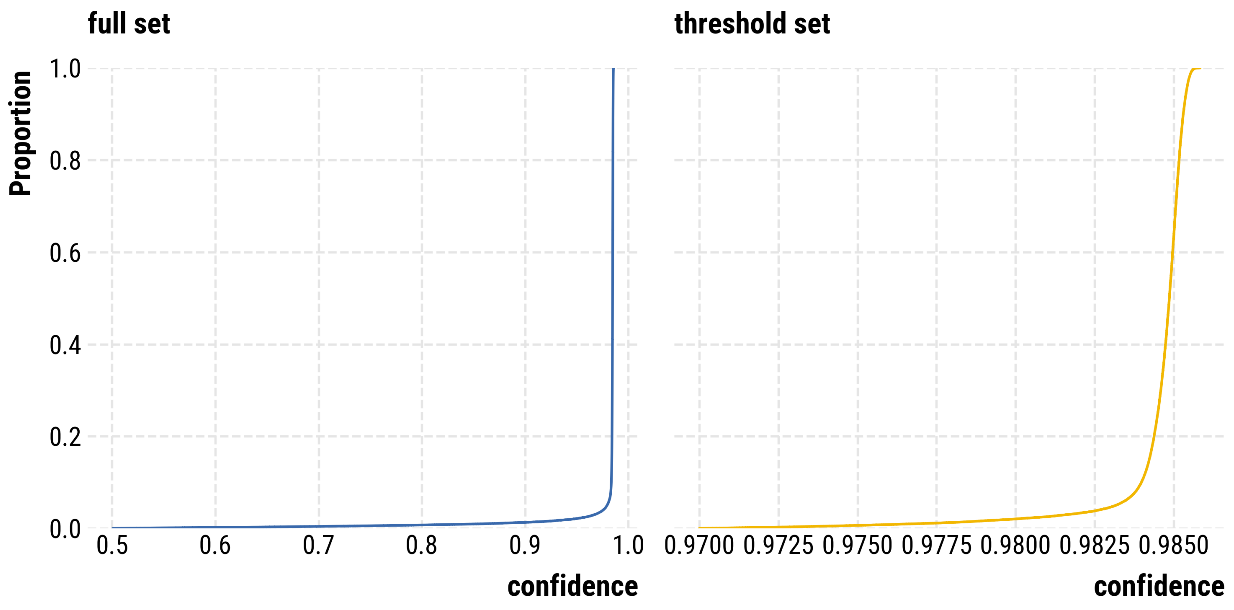

Figure 3: Distribution of author-developer-account prediction confidence scores. The left plot shows the distribution of all prediction confidence scores, while the right plot shows the distribution of prediction confidence scores for author-developer-account pairs with a confidence score greater than or equal to 0.97.

Thresholding the predictive model confidence at 0.97 resulted in a 3.2% (2911) reduction in the number of author-developer-account pairs (from an unfiltered total of 90086 author-developer-account pairs). This threshold was chosen to ensure a high level of confidence in the matches while retaining a large number of pairs for analysis.

7.3. Filtered Dataset Description for Article-Citation, Author-Position, and Author-Correspondence Analysis

Table 5: Counts of article-repository pairs, authors, and developers for research teams. Only includes research teams from article-repository pairs with a most recent commit no later than 90 days after publication and excludes research teams in the top 3% of total author sizes.

Category	**Subset**	**Article-Repository Pairs**	**Authors**	**Developers**
By Domain	Health Sciences	1167	6487	1462
	Life Sciences	1994	9586	2594
	Physical Sciences	18202	62061	22964
	Social Sciences	1441	5972	1913
By Document Type	preprint	3080	13082	4679
	research article	19502	70919	23993
	software article	222	1023	495
By Access Status	Closed	1190	5597	1835
	Open	21614	77234	26849
By Data Source	joss	86	416	281
	plos	2847	14331	3497
	pwc	19735	66217	24452
	softwarex	136	609	215
Total		**22804**	**80620**	**28290**

7.4. Additional Analysis of Code-Contributing Non-Authors

7.4.1. Sample Construction and Labeling

To better understand the nature and extent of contributions made by code-contributing non-authors in our dataset, we conducted a detailed analysis using a random sample of 200 individuals from our filtered dataset used in other analyses (Table 5). Our analysis combined qualitative labeling of contribution types with quantitative analysis of commit activity, additions, and deletions made by these contributors to their respective article-repository pairs.

7.4.1.1. Annotation Process.

Two independent annotators labeled each of the 200 code-contributing non-authors across three dimensions after completing two rounds of trial labeling on 20 cases to establish agreement. The final labeling criteria were:

Contribution Type:

- *"docs"*: Contributors who only modified documentation files (README, LICENSE, etc.) or made changes limited to code comments.
- *"code"*: Contributors who modified actual code files (.py, .R, .js, etc.) with substantive changes, or modified code support files (requirements.txt, pyproject.toml, package.json, etc.). Contributors who made both code and documentation changes were labeled as "code".
- *"other"*: Contributors whose changes did not fit the above categories, including those who committed to upstream forks or merged code without authoring it.

Author Matching Assessment:

- *"yes"*: Contributors who should have been matched to an author (missed classification).
- *"no"*: Contributors correctly classified as non-authors.
- *"unclear"*: Cases with insufficient information for determination.

Bot Account Detection:

- *"yes"*: Automated accounts (GitHub Actions, Dependabot, etc.).
- *"no"*: Human users.

After establishing criteria agreement, each annotator independently labeled 90 contributors, with the final sample of 200 created by combining both sets plus the 20 cases used for criteria development.

7.4.1.2. Quantitative Metrics.

For each code-contributing non-author, we collected commit activity data using the GitHub API contributor stats endpoint:

- **Number of Commits**: The total number of commits made by the code-contributing to the article-repository pair.
- **Number of Additions**: The total number of lines of code added by the code-contributing to the article-repository pair.
- **Number of Deletions**: The total number of lines of code deleted by the code-contributing to the article-repository pair.
- **Number of Total Repository Commits**: The total number of commits made to the article-repository pair, regardless of code-contributor.
- **Number of Total Repository Additions**: The total number of lines of code added to the article-repository pair, regardless of code-contributor.
- **Number of Total Repository Deletions**: The total number of lines of code deleted from the article-repository pair, regardless of code-contributor.

We additionally, calculated the absolute change for each code-contributor as the sum of additions and deletions, which provides a measure of the total impact of their contributions. Further, we normalized these metrics by the total number of commits, additions, deletions, and absolute changes made to the article-repository pair, regardless of code-contributor. This normalization allows us to compare the relative contribution of each code-contributing non-author to the overall contribution of the article-repository pair.

7.4.2. Results

We find that 39% (78) of code-contributing non-authors were correctly classified as non-authors, 30.5% (61) were unclear due to insufficient profile information, and 30.5% (61) appeared to be missed classifications that should have been matched to authors. Only 2 accounts (1%) were identified as bot accounts.

When broken out by contribution type, we find that:

- *"true non-authors" (n=78)*: 59 contributed code, 13 contributed documentation, and 4 contributed some other type of change
- *"missed classifications" (n=61)*: 49 contributed code, 12 contributed documentation, and 0 contributed some other type of change
- *"unclear" (n=61)*: 50 contributed code, 8 contributed documentation, and 3 contributed some other type of change

Table 6 and Table 7 present commit statistics for true non-authors and unclear cases, respectively. Among true non-authors making code contributions, the top quartile (75th percentile and above) contributed approximately 10% of total repository commits and 14.4% of absolute changes (additions + deletions). The unclear cases showed substantially higher contribution levels. Code contributors in this group made up 50.5% of total repository commits and 41.7% of repository absolute changes even at the 25th percentile.

Table 6: Commit statistics for code-contributing non-authors labeled as true non-authors. Statistics are calculated as the proportion of commits, additions, deletions, and absolute changes made by the code-contributing non-author to the total commits, additions, deletions, and absolute changes made to the article-repository pair, regardless of code-contributor.

	mean	std	min	25%	50%	75%	max
commit_stats	0.178	0.307	0.001	0.007	0.029	0.107	1.0
addition_stats	0.227	0.380	0.000	0.001	0.007	0.192	1.0
deletion_stats	0.193	0.358	0.000	0.000	0.006	0.149	1.0
abs_stats	0.222	0.379	0.000	0.001	0.010	0.144	1.0

Table 7: Commit statistics for code-contributing non-authors labeled as unclear. Statistics are calculated as the proportion of commits, additions, deletions, and absolute changes made by the code-contributing non-author to the total commits, additions, deletions, and absolute changes made to the article-repository pair, regardless of code-contributor.

	mean	std	min	25%	50%	75%	max
commit_stats	0.747	0.366	0.004	0.505	1.0	1.0	1.0
addition_stats	0.722	0.420	0.000	0.310	1.0	1.0	1.0
deletion_stats	0.737	0.417	0.000	0.722	1.0	1.0	1.0
abs_stats	0.733	0.404	0.000	0.417	1.0	1.0	1.0

We observed a notable pattern where very few true non-authors (n=4) were repository owners, while a substantial portion of unclear cases (n=30, 49%) owned the repositories they contributed to. This suggests many unclear contributors were likely primary code authors who could not be matched due to limited profile information. When excluding repository owners from the unclear group (Table 8), the median contribution drops to 34.6% of total commits and 12.7% of absolute changes, though this still represents substantial technical involvement.

Table 8: Commit statistics for code-contributing non-authors labeled as unclear, excluding repository owners. Statistics are calculated as the proportion of commits, additions, deletions, and absolute changes made by the code-contributing non-author to the total commits, additions, deletions, and absolute changes made to the article-repository pair, regardless of code-contributor.

	mean	std	min	25%	50%	75%	max
commit_stats	0.454	0.399	0.004	0.041	0.346	0.841	1.0
addition_stats	0.380	0.443	0.000	0.016	0.094	0.917	1.0
deletion_stats	0.486	0.492	0.000	0.003	0.431	0.999	1.0
abs_stats	0.392	0.440	0.000	0.011	0.127	0.919	1.0

Our analysis provides evidence that code-contributing non-authors represent a heterogeneous group with varying contribution levels. While defining "substantial" contribution worthy of authorship remains challenging, our findings reveal a clear mix of legitimate non-authors and potentially missed classifications, with both groups often contributing meaningful portions of repository commits and code changes.

However, our sample size of 200 limits generalizability to the full population of code-contributing non-authors. Additionally, the manual annotation process introduces potential subjectivity despite our established criteria, and our reliance on publicly available GitHub profiles may systematically underestimate contributions from developers with minimal profile information.

764 *7.5. Article Citation Linear Model Results*

Table 9: Article citations by code contributorship of research team. Generalized linear model fit with negative binomial distribution and log link function. Significant p-values are indicated with asterisks: p < 0.05 (*), p < 0.01 (**), p < 0.001 (***)

Variable	coef	P>|z|	[0.025	0.975]
const ***	**0.98**	**0.00**	**0.93**	**1.03**
Total Authors ***	**0.07**	**0.00**	**0.06**	**0.08**
Code-Contrib. Authors ***	**0.05**	**0.00**	**0.03**	**0.07**
Code-Contrib. Non-Authors	-0.00	0.56	-0.01	0.01
Years Since Publication ***	**0.39**	**0.00**	**0.38**	**0.40**

Table 10: Article citations by code contributorship of research team controlled by open access status. Generalized linear model fit with negative binomial distribution and log link function. Significant p-values are indicated with asterisks: p < 0.05 (*), p < 0.01 (**), p < 0.001 (***)

Variable	coef	P>|z|	[0.025	0.975]
const ***	**0.61**	**0.00**	**0.49**	**0.73**
Total Authors ***	**0.07**	**0.00**	**0.06**	**0.08**
Code-Contrib. Authors	0.05	0.26	-0.04	0.14
Code-Contrib. Non-Authors	0.00	0.91	-0.03	0.03
Years Since Publication ***	**0.38**	**0.00**	**0.37**	**0.39**
Is Open Access ***	**0.41**	**0.00**	**0.29**	**0.53**
Code-Contrib. Authors × Is Open Access	-0.00	0.98	-0.09	0.09
Code-Contrib. Non-Authors × Is Open Access	-0.00	0.82	-0.04	0.03

Table 11: Article citations by code contributorship of research team controlled by domain. Generalized linear model fit with negative binomial distribution and log link function. Significant p-values are indicated with asterisks: p < 0.05 (*), p < 0.01 (**), p < 0.001 (***)

Variable	coef	P>|z|	[0.025	0.975]
const ***	**0.89**	**0.00**	**0.75**	**1.02**
Total Authors ***	**0.07**	**0.00**	**0.06**	**0.08**
Code-Contrib. Authors	0.02	0.66	-0.08	0.12
Code-Contrib. Non-Authors	0.01	0.63	-0.04	0.07
Years Since Publication ***	**0.40**	**0.00**	**0.39**	**0.41**
Domain Life Sciences **	**-0.20**	**0.01**	**-0.35**	**-0.06**
Domain Physical Sciences *	**0.13**	**0.04**	**0.00**	**0.25**
Domain Social Sciences *	**-0.19**	**0.02**	**-0.35**	**-0.03**
Code-Contrib. Authors × Domain Life Sciences	0.07	0.29	-0.06	0.19
Code-Contrib. Authors × Domain Physical Sciences	0.02	0.75	-0.09	0.12
Code-Contrib. Authors × Domain Social Sciences	0.11	0.09	-0.02	0.24
Code-Contrib. Non-Authors × Domain Life Sciences	-0.04	0.29	-0.11	0.03
Code-Contrib. Non-Authors × Domain Physical Sciences	-0.02	0.55	-0.07	0.04
Code-Contrib. Non-Authors × Domain Social Sciences	-0.04	0.32	-0.11	0.04

Table 12: Article citations by code contributorship of research team controlled by article type. Generalized linear model fit with negative binomial distribution and log link function. Significant p-values are indicated with asterisks: $p < 0.05$ (*), $p < 0.01$ (**), $p < 0.001$ (***)

Variable	**coef**	**P>|z|**	**[0.025**	**0.975]**
const ***	**0.51**	**0.00**	**0.43**	**0.59**
Total Authors ***	**0.07**	**0.00**	**0.06**	**0.08**
Code-Contrib. Authors	-0.02	0.48	-0.07	0.03
Code-Contrib. Non-Authors **	**-0.03**	**0.00**	**-0.05**	**-0.01**
Years Since Publication ***	**0.40**	**0.00**	**0.39**	**0.41**
Article Type Research Article ***	**0.49**	**0.00**	**0.41**	**0.56**
Article Type Software Article ***	**-0.47**	**0.00**	**-0.73**	**-0.21**
Code-Contrib. Authors × Article Type Research Article **	**0.09**	**0.00**	**0.04**	**0.15**
Code-Contrib. Authors × Article Type Software Article	-0.07	0.30	-0.20	0.06
Code-Contrib. Non-Authors × Article Type Research Article ***	**0.04**	**0.00**	**0.02**	**0.06**
Code-Contrib. Non-Authors × Article Type Software Article	0.09	0.22	-0.06	0.23

7.6. Post-Hoc Tests for Coding vs Non-Coding Authors by Position

Table 13: Counts of Code-Contributing Authors ('Coding') as well as Total Authors by Position and Bonferroni Corrected p-values from Post-Hoc Binomial Tests. Significant p-values are indicated with asterisks: p < 0.05 (*), p < 0.01 (**), p < 0.001 (***).

Control	Subset	Position	Coding	Total	p
Domain	**Health Sciences**	First	745	1167	0.000^{***}
		Middle	255	4741	0.000^{***}
		Last	109	1156	0.000^{***}
	Life Sciences	First	1277	1992	0.000^{***}
		Middle	446	6289	0.000^{***}
		Last	252	1987	0.000^{***}
	Physical Sciences	First	12582	18116	0.000^{***}
		Middle	4992	51004	0.000^{***}
		Last	1207	17965	0.000^{***}
	Social Sciences	First	971	1438	0.000^{***}
		Middle	442	3649	0.000^{***}
		Last	139	1437	0.000^{***}
Article Type	**Preprint**	First	2118	3053	0.000^{***}
		Middle	912	8626	0.000^{***}
		Last	194	3050	0.000^{***}
	Research Article	First	13320	19438	0.000^{***}
		Middle	5069	56453	0.000^{***}
		Last	1467	19273	0.000^{***}
	Software Article	First	137	222	0.004^{**}
		Middle	154	604	0.000^{***}
		Last	46	222	0.000^{***}
Open Access Status	**Closed Access**	First	838	1184	0.000^{***}
		Middle	333	3647	0.000^{***}
		Last	76	1169	0.000^{***}
	Open Access	First	14737	21529	0.000^{***}
		Middle	5802	62036	0.000^{***}
		Last	1631	21376	0.000^{***}
Overall	**Overall**	First	15575	22713	0.000^{***}
		Middle	6135	65683	0.000^{***}
		Last	1707	22545	0.000^{***}

Counts of authors in Table 13 may differ slightly from counts in Table 5. Table 5 counts unique authors, while Table 13 counts unique author-document pairs (i.e., the same author may appear in multiple documents).

7.7. *Post-Hoc Tests for Coding vs Non-Coding Authors by Corresponding Status*

Table 14: Counts of Code-Contributing Authors ('Coding') as well as Total Authors by Corresponding Status and Bonferroni Corrected p-values from Post-Hoc Binomial Tests. Significant p-values are indicated with asterisks: $p < 0.05$ (*), $p < 0.01$ (**), $p < 0.001$ (***).

Control	**Subset**	**Is Corresponding**	**Coding**	**Total**	**p**
Domain	**Health Sciences**	Corresponding	542	3010	0.000***
		Not Corresponding	567	4054	0.000***
	Life Sciences	Corresponding	1031	4942	0.000***
		Not Corresponding	944	5326	0.000***
	Physical Sciences	Corresponding	2549	7238	0.000***
		Not Corresponding	16232	79847	0.000***
	Social Sciences	Corresponding	429	1515	0.000***
		Not Corresponding	1123	5009	0.000***
Article Type	**Preprint**	Corresponding	16	51	0.022*
		Not Corresponding	3208	14678	0.000***
	Research Article	Corresponding	4464	16458	0.000***
		Not Corresponding	15392	78706	0.000***
	Software Article	Corresponding	71	196	0.000***
		Not Corresponding	266	852	0.000***
Open Access Status	**Closed Access**	Corresponding	86	194	0.263
		Not Corresponding	1161	5806	0.000***
	Open Access	Corresponding	4465	16511	0.000***
		Not Corresponding	17705	88430	0.000***
Overall	**Overall**	Corresponding	4551	16705	0.000***
		Not Corresponding	18866	94236	0.000***

Counts of authors in Table 14 may differ slightly from counts in Table 5. Table 5 counts unique authors, while Table 14 counts unique author-document pairs (i.e., the same author may appear in multiple documents).

7.8. *Filtered Dataset Description for h-Index Analysis*

Table 15: Counts of Total Authors, n Any Coding Authors, n Majority Coding Authors, and n Always Coding Authors by Common Domain, Document Type, and Author Position. Authors are only included if they have three or more publications within our dataset and are associated with no more than three developer accounts, with each association having a predicted model confidence of at least 97%.

Category	**Subset**	**Total Authors**	**Any Code**	**Majority Code**	**Always Code**
By Commmon Domain	Health Sciences	1507	339	196	82
	Life Sciences	1440	351	236	129
	Physical Sciences	49430	14753	7951	3720
	Social Sciences	1304	276	219	178
By Document Type	Preprint	29038	9255	4828	2151
	research article	24265	6419	3657	1830
	software article	378	45	117	128
By Author Position	First	11459	1671	4864	3249
	last	10208	2260	550	186
	middle	32014	11788	3188	674
Total		**53681**	**15719**	**8602**	**4109**

7.9. h-Index Linear Model Results

Table 16: Code-contributing authors h-index by coding status. Generalized linear model fit with Gaussian distribution and log link function. Significant p-values are indicated with asterisks: p < 0.05 (*), p < 0.01 (**), p < 0.001 (***).

Variable	coef	P>|z|	[0.025	0.975]
const ***	**3.18**	**0.00**	**3.17**	**3.19**
Works Count ***	**0.00**	**0.00**	**0.00**	**0.00**
Any Coding ***	**-0.32**	**0.00**	**-0.34**	**-0.31**
Majority Coding ***	**-0.76**	**0.00**	**-0.79**	**-0.73**
Always Coding ***	**-0.96**	**0.00**	**-1.01**	**-0.91**

Table 17: Code-contributing authors h-index by coding status controlled by most freq. author position. Generalized linear model fit with Gaussian distribution and log link function.Significant p-values are indicated with asterisks: p < 0.05 (*), p < 0.01 (**), p < 0.001 (***).

Variable	coef	P>|z|	[0.025	0.975]
const ***	**2.36**	**0.00**	**2.30**	**2.42**
Works Count ***	**0.00**	**0.00**	**0.00**	**0.00**
Any Coding ***	**0.16**	**0.00**	**0.07**	**0.24**
Majority Coding	-0.05	0.19	-0.12	0.03
Always Coding ***	**-0.22**	**0.00**	**-0.31**	**-0.14**
Common Author Position Last ***	**1.06**	**0.00**	**1.00**	**1.13**
Common Author Position Middle ***	**0.75**	**0.00**	**0.69**	**0.82**
Any Coding × Common Author Position Last ***	**-0.31**	**0.00**	**-0.40**	**-0.23**
Any Coding × Common Author Position Middle ***	**-0.47**	**0.00**	**-0.55**	**-0.38**
Majority Coding × Common Author Position Last ***	**-0.37**	**0.00**	**-0.47**	**-0.28**
Majority Coding × Common Author Position Middle ***	**-0.62**	**0.00**	**-0.71**	**-0.54**
Always Coding × Common Author Position Last ***	**-0.38**	**0.00**	**-0.52**	**-0.23**
Always Coding × Common Author Position Middle ***	**-0.51**	**0.00**	**-0.64**	**-0.38**

Table 18: Code-contributing authors h-index by coding status controlled by most freq. domain. Generalized linear model fit with Gaussian distribution and log link function.Significant p-values are indicated with asterisks: $p < 0.05$ (*), $p < 0.01$ (**), $p < 0.001$ (***).

Variable	coef	P>|z|	[0.025	0.975]
const ***	**3.31**	**0.00**	**3.27**	**3.35**
Works Count ***	**0.00**	**0.00**	**0.00**	**0.00**
Any Coding ***	**-0.37**	**0.00**	**-0.46**	**-0.29**
Majority Coding ***	**-1.46**	**0.00**	**-1.66**	**-1.26**
Always Coding ***	**-1.19**	**0.00**	**-1.55**	**-0.82**
Common Domain Life Sciences ***	**0.11**	**0.00**	**0.06**	**0.16**
Common Domain Physical Sciences ***	**-0.14**	**0.00**	**-0.18**	**-0.10**
Common Domain Social Sciences ***	**-0.16**	**0.00**	**-0.22**	**-0.10**
Any Coding × Common Domain Life Sciences	0.07	0.20	-0.04	0.19
Any Coding × Common Domain Physical Sciences	0.05	0.23	-0.04	0.14
Any Coding × Common Domain Social Sciences	0.01	0.87	-0.13	0.15
Majority Coding × Common Domain Life Sciences ***	**0.83**	**0.00**	**0.60**	**1.07**
Majority Coding × Common Domain Physical Sciences ***	**0.72**	**0.00**	**0.51**	**0.93**
Majority Coding × Common Domain Social Sciences ***	**0.77**	**0.00**	**0.50**	**1.03**
Always Coding × Common Domain Life Sciences	0.27	0.20	-0.14	0.69
Always Coding × Common Domain Physical Sciences	0.23	0.23	-0.14	0.60
Always Coding × Common Domain Social Sciences	0.30	0.17	-0.13	0.73

Table 19: Code-contributing authors h-index by coding status controlled by most freq. article type. Generalized linear model fit with Gaussian distribution and log link function.Significant p-values are indicated with asterisks: $p < 0.05$ (*), $p < 0.01$ (**), $p < 0.001$ (***).

Variable	coef	P>|z|	[0.025	0.975]
const ***	**3.09**	**0.00**	**3.08**	**3.10**
Works Count ***	**0.00**	**0.00**	**0.00**	**0.00**
Any Coding ***	**-0.29**	**0.00**	**-0.32**	**-0.27**
Majority Coding ***	**-0.76**	**0.00**	**-0.80**	**-0.72**
Always Coding ***	**-0.98**	**0.00**	**-1.06**	**-0.90**
Common Article Type Research Article ***	**0.18**	**0.00**	**0.17**	**0.20**
Common Article Type Software Article ***	**0.22**	**0.00**	**0.12**	**0.33**
Any Coding × Common Article Type Research Article *	**-0.03**	**0.05**	**-0.06**	**-0.00**
Any Coding × Common Article Type Software Article	0.17	0.10	-0.04	0.37
Majority Coding × Common Article Type Research Article	0.01	0.80	-0.05	0.06
Majority Coding × Common Article Type Software Article ***	**0.38**	**0.00**	**0.20**	**0.56**
Always Coding × Common Article Type Research Article	0.00	0.97	-0.10	0.10
Always Coding × Common Article Type Software Article **	**0.37**	**0.00**	**0.16**	**0.58**

7.10. Analysis of Project Duration and Percentage Code-Contributors Who Are Authors

In our pre-registered analysis plan (https://osf.io/fc74m), we originally hypothesized about the relationship between project duration and authorship recognition. Specifically, we posited that sustained technical engagement and scientific recognition might be meaningfully related, with longer project durations potentially leading to higher rates of code-contributor authorship. We saw repository histories as providing a unique opportunity to examine this relationship, leading us to hypothesize that projects with longer commit durations would be associated with higher percentages of developers receiving authorship recognition (pre-registered as H2).

However, our analysis found no evidence to support this hypothesis. When examining the relationship between a repository's commit duration and the percentage of developers who receive authorship recognition, we found no significant correlation (r = 0.00, p = n.s.). This suggests that the length of time a project has been in development has no meaningful relationship with the proportion of developers who are recognized as authors.

We ultimately decided to move this analysis to the appendix for two key methodological reasons. First, our approach of using repository-level commit duration as a proxy for individual contribution patterns proved too coarse-grained. A more precise analysis would need to examine individual-level contribution durations and patterns rather than overall project length. Second, our method did not account for the varying levels of contribution that different developers make to a repository. Simply correlating overall project duration with authorship rates fails to capture the nuanced ways that sustained, meaningful technical contributions might influence authorship decisions.

These limitations suggest potential directions for future work that could more rigorously examine the relationship between long-term technical engagement and scientific recognition. Such work might benefit from more granular analysis of individual contribution patterns, perhaps incorporating measures of contribution significance and sustainability rather than just temporal duration.