Code Contribution and Credit in Science

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 investigate these concepts 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 —investigating 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 ~4.2% 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 than non-coding colleagues, even when controlling for publication count, author position, domain, and article type. These results suggest a disconnect between software contributions and credit, highlighting important implications for institutional reward structures and science policy.

1. Introduction

Software and collaboration are two essential ingredients in contemporary science but are often investigated separately: Collaboration is typically studied using bibliometric data to understand, for example, who works with whom (Newman et al., 2014), who receives what credit (Shen and Barabási, 2014), and to what effect (Ahuja, 2000). Similarly, scientific software is often quantitatively 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 affects 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 complement traditional bibliometric indicators. We first build a dataset (`rs-graph-v1`) of 138,596 paired research articles and code repositories — giving us a verified connection between software and the published literature that it supports. 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 commit histories from the public software repositories' to test three hypotheses.

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

We identify several patterns that distinguish code contributions from formal scientific recognition. We use a filtered set of one-to-one article-repository pairs (removing outliers for research team size, repository commit duration, and having at least one citation) to show that 28.6% (n=6,529) 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 (on average, a 4.2% increase in citations per code-contributing author), but these effects become statistically non-significant when controlling for domain, article type, and open access status. We find that 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 even when controlling for publication count, and author's most common author position (first, middle, or last), domain (Physical Sciences, Health Sciences, Life Sciences, or Social Sciences), and article type (preprint, research article, or software article) [1].

The remainder of this paper proceeds as follows. First, we review related work regarding software development and the recognition of code contributors in scientific credit systems. In doing so, we motivate 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 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 can produce.

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, where established scientists receive more attention for work of similar importance or value than their less established peers (Merton, 1968). In quantitative work, credit is most often established using 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 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 in collaborative settings.

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 development and long-term maintenance describe inequalities in credit, recognition, and career advancement for software work (Muna et al., 2016). Much of the existing literature on the role and importance of software in science is either ethnographic (Howison and Herbsleb, 2011, Paine and Lee

[1] We use the phrase "most common" to mean most frequent, and in the case of a tie, their most recent. That is, if an author has been first author on four publications in our dataset, middle author on two publications, and has never been last author, they are considered a "first author" for the purposes of our analysis. We discuss the limitations of this approach in the Section 5.

(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 quantitative research. In this paper, we build a unique dataset that allows 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 hypotheses 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 produce higher-impact scientific work (Franceschet and Costantini, 2010; Larivière et al., 2014). Research in empirical software studies similarly finds 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 reproducability remain gold standards (Milewicz and Raybourn, 2018).

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 citation practices in collaborative research, we propose that:

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 that can be contemporarily studied through contribution role taxonomies like CRediT (Larivière et al., 2016). In this literature, 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). While these contribution patterns are often inferred from self-reported roles (e.g., CRediT) or disciplinary conventions, the linking of publications to version-controlled software repositories provides a novel window into the actual coding contributions associated with scientific papers. Given the established tendency for first authors to undertake primary research execution, including technical development, we anticipate that this pattern will be directly observable via the commit histories of scientific software repositories. 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.

Furthermore, studies that use contribution taxonomies like CRediT reveal that first authors and corresponding authors, while occasionally the same individual, often take on distinct responsibilities (Birnbaum

et al., 2023). Corresponding authors traditionally bear responsibility for the integrity, archiving, and long-term availability of research artifacts and underlying data (Teunis et al., 2015; Sauermann and Haeussler, 2017; da Silva et al., 2013; Bhandari et al., 2014). As software becomes an increasingly critical, complex, and integral research artifact in many disciplines, it logically follows that stewardship of these software components—encompassing their maintenance, documentation, and accessibility—should align with the responsibilities of being a corresponding author. Given prior findings about the distribution of responsibilities within research teams and the expectation of stewardship of created artifacts, we expect repository records to reflect similar patterns of engagement with software development:

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 dedicate significant effort to its development often face systemic hurdles in receiving formal scientific credit. Their contributions may be relegated to acknowledgment sections rather than rewarded with authorship credit. Further, the scholarly practice of software citation remains inconsistent, frequently undervaluing crucial maintenance and extension work (Carver et al., 2022; Philippe et al., 2019; Lamprecht et al., 2020; Katz et al., 2020; Smith et al., 2016, Weber and Thomer (2014)). The h-index, a widely used proxy for an individual's cumulative scientific impact, is derived from an individual's record of formally authored publications and the citations these publications receive (Hirsch, 2005). Consequently, if substantial time and intellectual effort are invested in software development that does not consistently translate into formal authorship on associated research papers, or if the software outputs themselves are not robustly and formally cited in a way that accrues to the individual developer, then the primary activities that build a researcher's h-index are effectively diminished or bypassed.

This creates a structural misalignment where contributions essential for scientific advancement (i.e., software development and maintenance) may not adequately capture and could even detract from time spent on activities that bolster traditional bibliometric indicators of individual success. While collaborative software development may yield short-term benefits through increased citations on individual papers (as suggested by H1), researchers specializing in code development may face long-term career disadvantages as their expertise becomes increasingly divorced from traditional publication pathways that drive academic recognition and advancement. Based on these challenges in the recognition and citation of software contributions and their potential impact on h-index accumulation, we hypothesize:

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

3. Data and Methods

Our analysis examines the relationship between software contributions and scientific credit 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 from ArXiv with software 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 in our corpus, we resolved the source DOI via Semantic Scholar to ensure we captured its latest version and then extracted publication metadata and author metrics through OpenAlex. Finally, we collected information

about each repository and the repository's contributors via the GitHub API. A data collection and processing workflow diagram is available in Figure 1. All data was collected between October and November 2024 [2].

Articles collected from JOSS and SoftwareX were labeled as "software articles," articles obtained from PLOS were labeled as "research articles," and articles from Papers with Code were labeled as either "preprint" or "research article" based on their document type from OpenAlex. While Papers with Code data is largely tied to ArXiv preprints, because of our DOI resolution process, we were able to identify a subset of these articles that had been published in a journal and were thus labeled as "research articles" by OpenAlex [3]. Article domain classifications (e.g., Health Sciences, Life Sciences, Physical Sciences, and Social Sciences) were obtained from OpenAlex. While each article is associated to multiple "topics", we utilize the primary topic and its associated domain for our analyses [4].

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 such as when 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). When exact matching fails, there is typically high semantic overlap between an author's name and developer account details (i.e., username, name, and email) that transformer models 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 a binary F1 score of 0.944, with 0.938 precision and 0.95 recall (positive = "match"). A detailed comparison of models and feature sets, and an evaluation of model performance across non-JOSS author-developer pairs is made available in the Appendix. Applying our model across all article-repository pairs yielded a large-scale dataset linking scientific authors and code contributors.

Our complete dataset, named the "Research Software Graph" (`rs-graph-v1`), contains 163,292 article-repository pairs (2,529 from JOSS, 6,350 from PLOS, 153,795 from Papers with Code, and 618 from SoftwareX), 157,307 unique articles (2,529 from JOSS, 6,350 from PLOS, 147,863 from Papers with Code, and 598 from SoftwareX), and 153,588 unique repositories (2,516 from JOSS, 6,245 from PLOS, 144,492 from Papers with Code, and 612 from SoftwareX). However, 24,696 article-repository pairs form many-to-many relationships in which the same article is linked to multiple repositories or the same repository is linked to multiple articles. The median number of repositories per article is 1 (95th percentile: 1, 97th percentile: 2, 99th percentile: 2). The median number of articles per repository is 1 (95th percentile: 1, 97th percentile: 2, 99th percentile: 2).

To ensure that our analysis focuses on clear, unambiguous relationships between articles and repositories, prior to analysis we utilize a partial `rs-graph-v1` dataset which filters out any articles or repositories associated to multiple article-repository pairs. This filtering step removes 24,696 article-repository pairs. As shown in Table 1, the complete one-to-one article-repository dataset contains 138,596 unique article-repository pairs spanning multiple domains and publication types (2,336 unique articles, repositories, and pairs from

[2]The Journal of Open Source Software (JOSS) is available at https://joss.theoj.org/. SoftwareX articles are available at https://www.sciencedirect.com/journal/softwarex/ and SoftwareX repositories are available at https://github.com/ElsevierSoftwareX/. Public Library of Science (PLOS) is available at https://plos.org/ and article data is available via the `allofplos` Python package (https://github.com/PLOS/allofplos/). Papers with Code is available at https://paperswithcode.com/ and data for links between papers and code is available at https://paperswithcode.com/about/. Documentation for the Semantic Scholar API is available at https://api.semanticscholar.org/, documentation for the OpenAlex API is available at https://docs.openalex.org/, and documentation for the GitHub API is available at https://docs.github.com/en/rest/.

[3]We recognize that Papers with Code (arXiv) preprints likely include some number of software articles but without a clear and consistent mechanism to identify these articles, we are unable to label them as such. We discuss this limitation further in Section 5.

[4]OpenAlex documentation for concepts, topics, fields, and domains is available at https://help.openalex.org/hc/en-us/articles/24736129405719-Topics/

JOSS, 6,090 from PLOS, 129,615 from Papers with Code, and 555 from SoftwareX). Additionally the one-to-one dataset includes information for 295,806 distinct authors and 152,170 developer accounts. The one-to-one dataset includes 90,086 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. The complete `rs-graph-v1` dataset (https://doi.org/10.7910/DVN/KPYVI1), the trained model (https://huggingface.co/evamxb/dev-author-em-clf), and a supporting inference library for using the model: sci-soft-models, 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 Complete, Unfiltered Dataset.

Category	Subset	Article-Repository Pairs	Authors	Developers
By Domain	Health Sciences	5,176	26,022	7,277
	Life Sciences	7,727	31,613	12,123
	Physical Sciences	116,597	240,536	130,601
	Social Sciences	8,839	29,239	14,023
By Document Type	preprint	72,177	170,301	87,311
	research article	63,528	173,183	78,935
	software article	2,891	9,294	12,868
By Access Status	Closed	5,740	23,668	9,352
	Open	132,856	286,874	147,831
By Data Source	joss	2,336	7,105	11,362
	plos	6,090	30,233	8,784
	pwc	129,615	262,889	134,926
	softwarex	555	2,244	1,628
Total		**138,596**	**295,806**	**152,170**

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, to ensure that we aren't analyzing "data repositories" (i.e. GitHub repositories which only store CSV, Parquet, or other dataset related file formats), we filter out any article-repository pairs that don't have any programming language files [5]. Next, to ensure that the research has received a basic level of engagement from the scientific community, we remove any article-repository pairs which do not have at least one citation. We then require that repository commit activity stop before 90 days past the article publication date. 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 11 authors, the 97th percentile for research team size. Finally, we filter out any predicted author-developer pairs with a confidence score of less than 0.97 in order to improve robustness of our downstream analysis [6]. This filtering process results in a

[5] We use the list of programming langauges used by the GitHub platform via the Linguist software, and the GitHub API to retrieve the bytes of code per-language within each repository. We remove article-repository pairs which have a non-zero number of bytes of code for programming language files.

[6] Figure 5 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

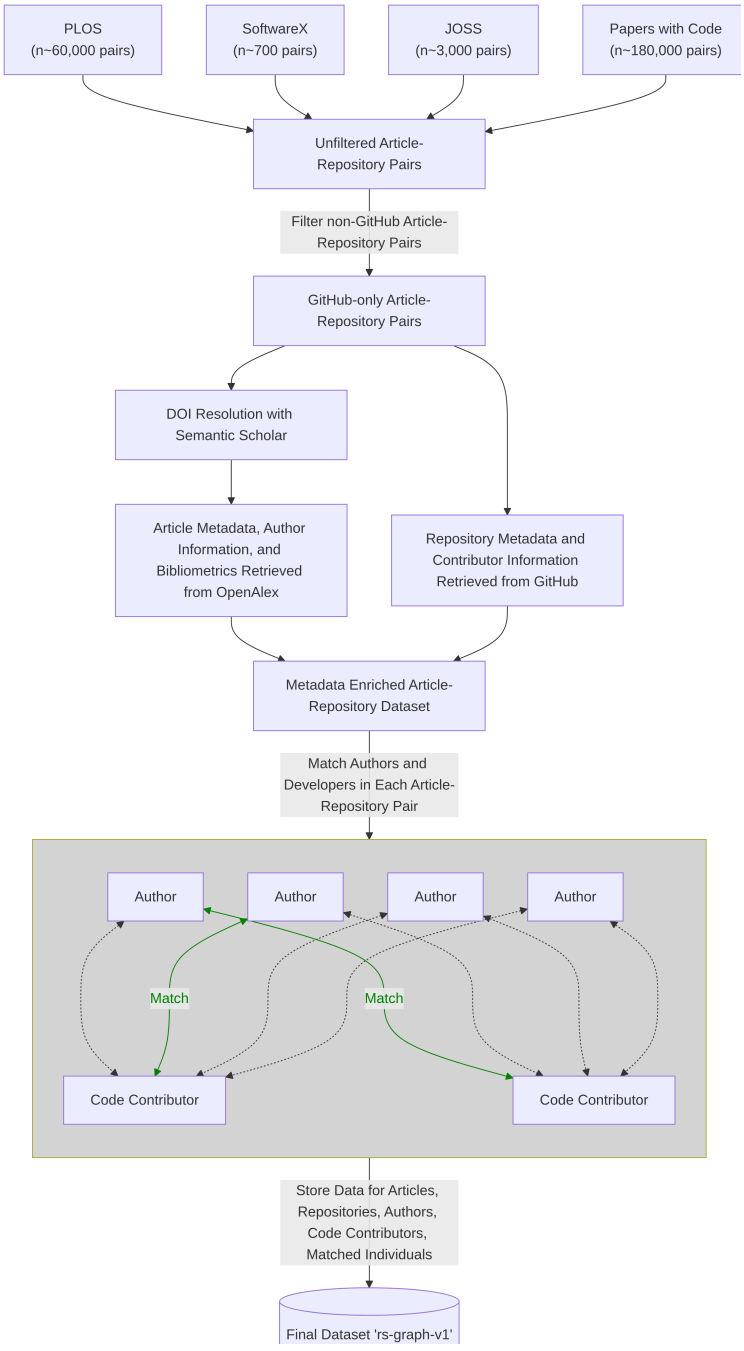

Figure 1: Data Gathering and Processing Workflow. We begin by gathering article-repository pairs from PLOS, JOSS, SoftwareX, and Papers with Code, retaining only pairs where the source code repository is hosted on GitHub. For each article-repository pair, we retrieve the DOI for the most recent version of the article via the Semantic Scholar API, then obtain the article's metadata, author information, and bibliometric data from OpenAlex. In parallel, we collect the source code repository's metadata and contributor information using the GitHub API. After retrieving both article and repository information, we apply our trained author-developer matching model across all possible combinations of article authors and repository contributors for each article-repository pair. Finally, we store all retrieved information and the matched author-developer pairs in a SQLite database. Article-repository pair counts are approximate because there are no snapshots of these databases at a single point in time. The estimates are based on counts from each data source taken in October 2025.

dataset of 19,900 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 code to associated repositories[7], non-code-contributing authors (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 4.8 ± 1.8 total authors, with 1.0 ± 0.7 code-contributing authors and 3.8 ± 1.9 non-code-contributing authors. Beyond the author list, papers averaged 0.5 ± 1.8 code-contributing non-authors. Figure 2 details these distributions by domain, article type, and open access status, as well as overall across filtered analysis set [8].

Perhaps most striking is our finding that 5,694 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 ± 3.0 un-matched code contributors per paper. To better understand the nature of these developers contributions, we conducted an analysis using a random sample of 200 code-contributing non-authors (see Section 7.4 for complete methodology).

Our analysis revealed three distinct groups: ~39% (n=78) represent true non-authors (individuals who likely should not be matched to an author), ~30.5% (n=61) appear to be missed classifications (individuals who likely should have been matched to authors), and the remaining ~30.5% (n=61) were unclear due to limited profile information [9]. Additional qualitative error analysis was conducted across the set of 61 likely authors and our findings are available in the appendex (Section 7.4.3). The majority of contributors across all groups made code changes rather than just documentation updates. Among true non-authors, ~77.6% (n=59) contributed code, with the top 25th percentile of these contributors contributing ~10.7% of total repository commits and ~14.4% 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 ~34.6% of repository commits and ~12.7% of absolute changes. Due to the limited size of our sample (n=200 code-contributing non-authors), 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 suggest systematic gaps in how scientific software development work is credited.

When examining these patterns over time and across different team sizes (Figure 3), we found that the number of code-contributing authors and unrecognized contributors has remained relatively stable. This 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

data (87,175 author-developer pairs) as only 2,911 author-developer-account pairs have a confidence less than 0.97 in the whole unfiltered dataset.

[7]We define code-contributing authors as those who have at least one code commit to the repository associated with the article-repository pair.

[8]A table with the means and standard deviations for each distribution and subset of team composition is provided in the appendix in Table 6.

[9]While annotation for the model training and evaluation set was done using primarily simple text information (e.g., usernames, emails, etc.), annotation conducted during the qualitative analysis of code-contributing non-authors included a more in-depth review of public profiles including any linked websites or social media profiles. This was done to gain a true best estimate for the "true", "missed", and "unclear" classifications.

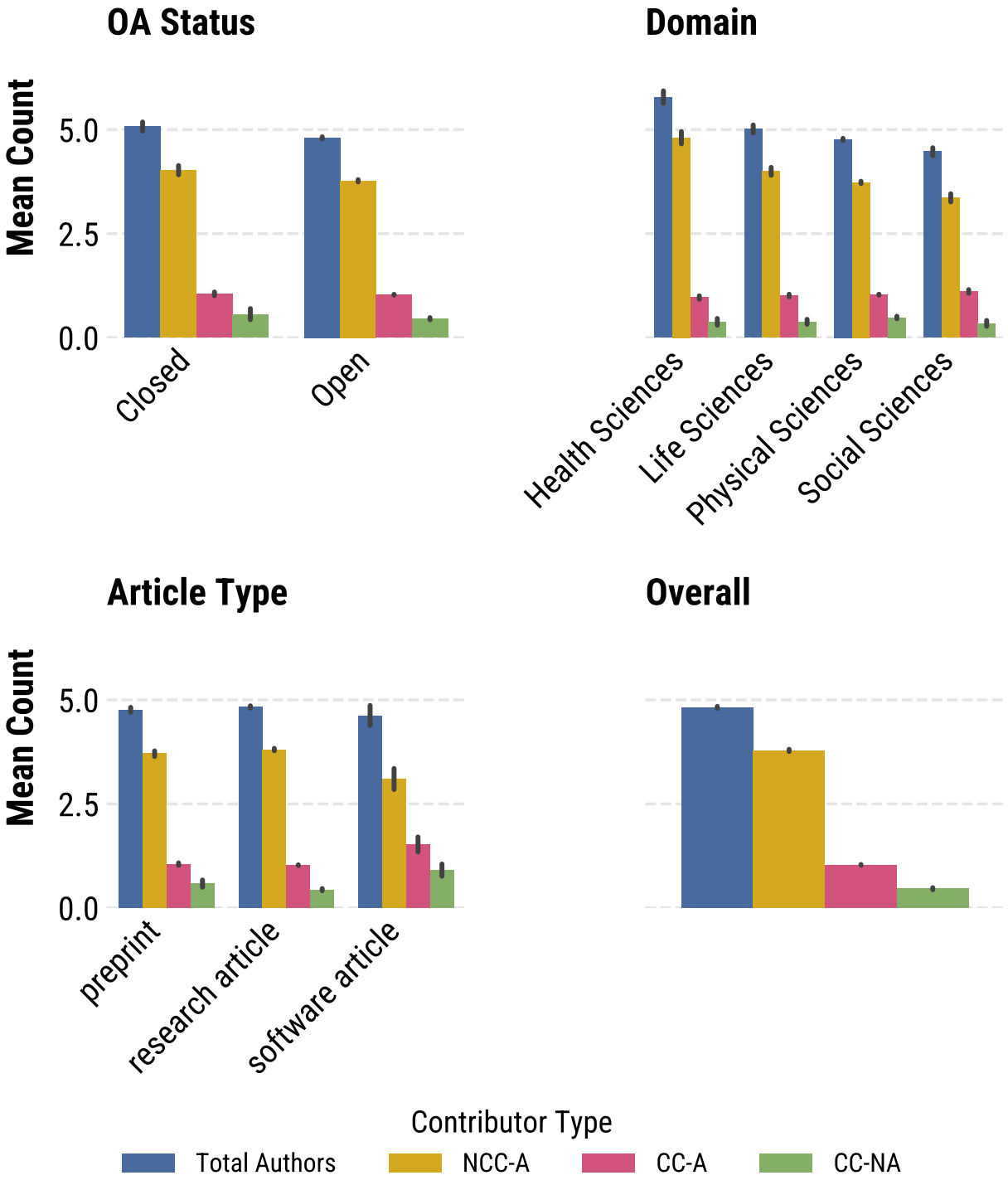

Figure 2: Mean of Non-Code-Contributing Authors (NCC-A), Code-Contributing Authors (CC-A), and Code-Contributing Non-Authors (CC-NA) Research Team Members by Domain, Article Type, and Open Access Status. Only includes research teams from article-repository pairs with repositories that have programming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

size grows, indicating that larger research teams do not necessarily adopt more inclusive authorship practices for code contributors despite representing broader collaborative efforts.

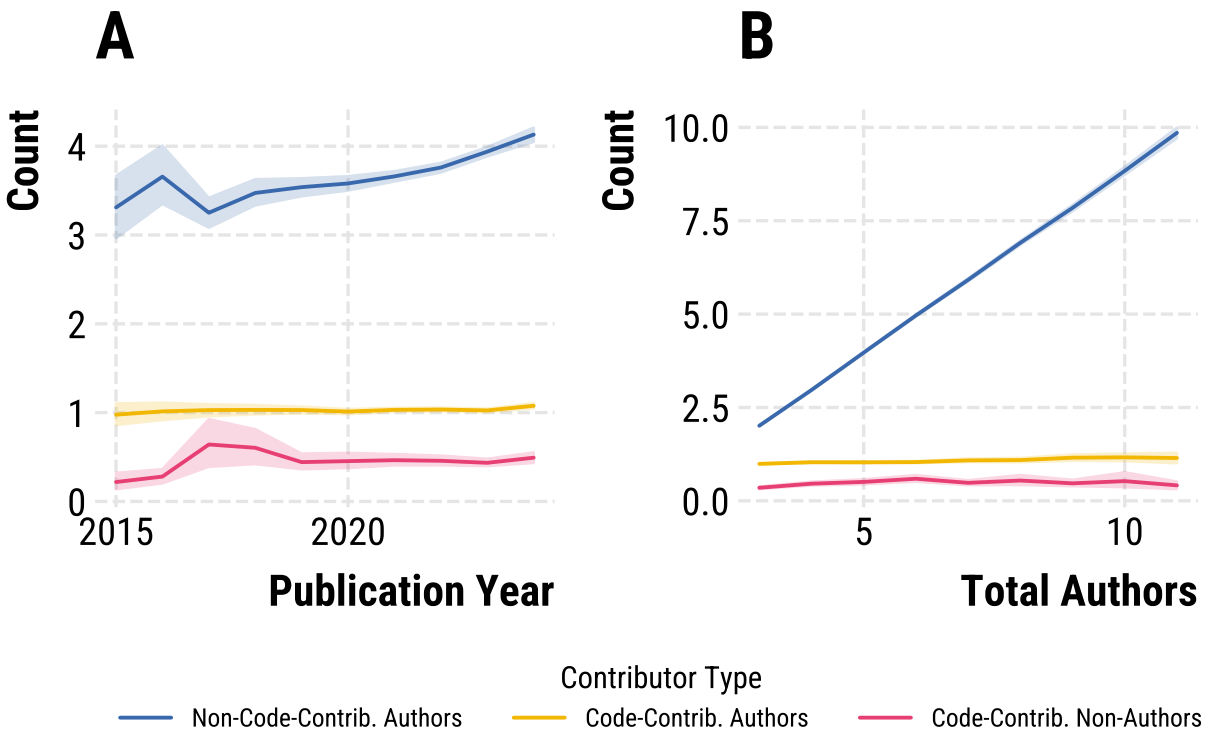

Figure 3: Average number of contributors per article, by contribution type, along with A) the year the article was published and B) the total number of authors included in the article. Only includes research teams from article-repository pairs with repositories that have programming language files, a 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 (*H1*).

Using our filtered dataset of article-repository pairs (Table 5), we conducted multiple generalized linear regression analyses to examine these relationships while controlling for various factors. Each generalized linear regression model included controls for the total number of authors listed on the publication and the number of years since the date of publication (as a decimal). Without additional controls for domain, open access, or article type (Table 10), our analysis revealed a modest positive association between the number of code-contributing authors and article citations, with each code-contributing author associated with, on average, a ~4.2% increase in article citations ($p < 0.001$).

When controlling for article type (Table 13), we observed divergent patterns between preprints and research articles. For preprints, each code-contributing non-author was associated with a statistically significant ~3.0% 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 **fail to reject** our hypothesis (*H1*) that "the number of individuals contributing code to a publication's associated repository positively correlates with the article's citation count." It is important to note that, the relationship between contributors and citations is statistically significant, but these effects are modest in magnitude, and differ 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 we originally stated in our hypothesis. We return to this finding in the discussion and limitations sections that follow.

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 (*H2a*).

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 [10]. Following these significant associations, we examined the specific proportions across positions (Table 14): 69.8% of first authors contributed code to their projects, compared to only 9.7% 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 ***fail to reject*** 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 (69.8%) is significantly higher than the proportion of both middle authors (9.7%) 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 (*H2b*)

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 29.7% of corresponding authors and 20.6% of non-corresponding authors contributing code to their projects. Further examination (Table 15) showed that this pattern holds across nearly all conditions,

[10] Community practices and norms around authorship order vary across disciplines and over time. One particular changing practice is the adoption of shared first authorship, where two or more authors are designated as having contributed equally to the work. However, after all filtering required for analysis was complete, none of the remaining articles included multiple first authors.

with only one 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 (29.7%) did not significantly differ from the proportion among non-corresponding authors (20.6%). This pattern of similar proportions remained consistent across most studied conditions, with a single exception in closed-access publications which we believe is due to the relatively small sample size available for testing (n=194).

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 predictive model confidence of less than 0.97. Table 16 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 4 shows the distribution of author h-indices across these coding frequency groups, grouped by author position, publication type, and research domain.

Using multiple generalized linear regressions, and while controlling for an author's total number of published works, we find a consistent and statistically significant negative relationship between code contribution frequency and h-index across multiple controls. Our initial analysis, controlled only by an author's publication count (Table 17) 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 18), 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 who always coded 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 19), 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 20) 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.

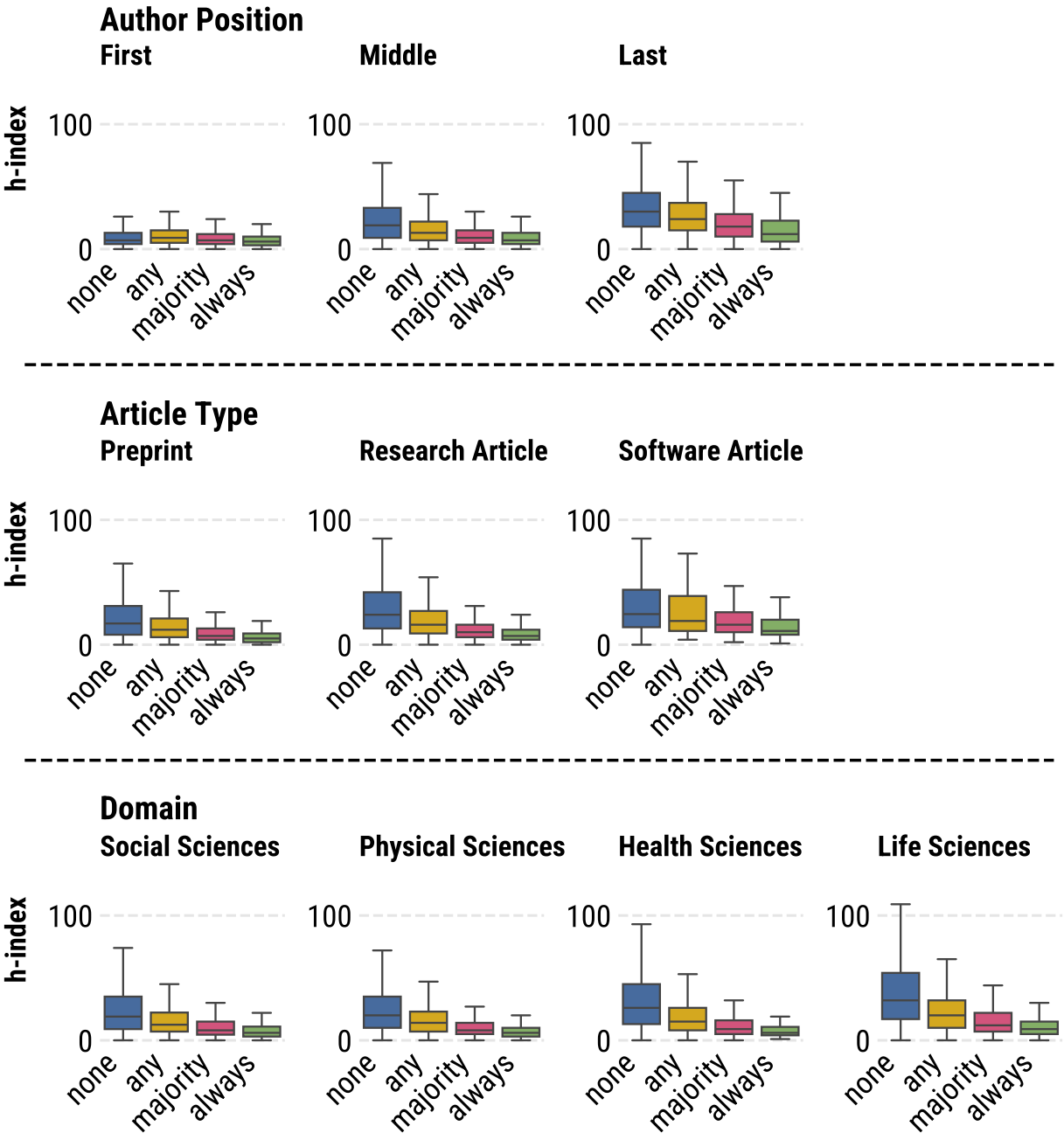

Figure 4: Distribution of author h-index by coding frequency across three key publication factors. Results are grouped by each author's most frequent: (1) position in publication bylines (first, middle, or last), (2) publication type (preprint, research article, or software article), and (3) research 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%.

Based on these findings, we ***fail to reject*** 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 our models include publication count as an input feature, suggesting that these h-index reductions persist even when accounting for total research output.

5. Discussion

Our analysis reveals significant disparities in the recognition of software contributions to scientific research, with ~28.6% (n=6,529) 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. 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 are significantly more likely to contribute code with 69.8% of all first authors in our dataset contributing code. Middle and last authors, meanwhile, were statistically significantly less likely to contribute code, with only 9.7% 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 29.7% 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 (~4.2% 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 code 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 more than 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 to consider 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.

Further, while our data processing workflow began with ~60,000 possible article-repository pairs from PLOS, ~3,000 from JOSS, ~700 from SoftwareX, and ~180,000 from Papers with Code for a possible total of ~243,700 [11], `rs-graph-v1` contained a total of 163,292 article-repository pairs. Many of the possible article-repository pairs were filtered out due to the linked repository not being hosted on GitHub, GitHub repositories not being found (either private, non-existent, or moved), or SemanticScholar or OpenAlex not indexing the article.

Our labeling of article types (software article, research article, preprint) was based on the data source (PLOS, JOSS, SoftwareX, Papers with Code) and in the case of Papers with Code articles, our DOI resolution process and the document type available from OpenAlex. This approach may misclassify certain articles, especially those from Papers with Code (arXiv). One potential alternative approach would involve classification of the repository itself following the recommendations of Hasselbring et al. (2025) in breaking down repositories by their role in research (e.g., "Modeling, Simuluation, and Data Analysis", "Technology Research Software", and "Research Infrastructure Software"). This classification would allow us to investigate not only the differences of "software papers" vs "research articles" and "preprints" (which we believe would both typically be paired with "Modeling, Simulation, and Data Analysis" repositories), but the *purpose* of the code as it relates to the research. However, there is currently no established automated method for performing this classification at scale.

Similarly, our simplification of author positions, domains, and article types to each author's "most common" (most frequent, with ties broken by most recent) introduces potential biases. This reduction may obscure the diversity of an author's contributions across different contexts, particularly for interdisciplinary researchers or those with varied roles in different projects. Further, these labels are created from metadata for articles only within our dataset. That is, even though a researcher may have dozens of articles, their "most common" author position, domain, and article type was determined with data for article-repository pairs. This inherently biases the dataset towards research teams who, as a collective, frequently create and share software and code as a part of their research process. While this approach was necessary for managing the complexity of our analysis, it may not fully capture the nuances of individual research careers.

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

[11] Article-repository pair counts are approximate because there are no snapshots of these databases at a single point in time. The estimates are based on counts from each data source taken in October 2025.

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 but 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. However, as discussed further in the Appendix (Section 7.1.3), our dataset is relatively diverse, with the median preprint-repository pair having a commit duration (the number of days between the repository's creation and the repository's most recent commit) of 53 days, research article-repository pairs having a median commit duration of 114 days, and software article-repository pairs having a median commit duration of 282 days. This diversity in commit durations suggests that our dataset contains a range of development practices, including both some "code dumps," and year (and multi-year) long projects.

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 11 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 must address introducing 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 .

Bhandari, M., Guyatt, G.H., Kulkarni, A.V., Devereaux, P.J., Leece, P., Bajammal, S., Heels-Ansdell, D., Busse, J.W., 2014. Perceptions of authors' contributions are influenced by both byline order and designation of corresponding author. Journal of Clinical Epidemiology 67, 1049–1054. URL: https://www.sciencedirect.com/science/article/pii/S0895435614001267, doi:https://doi.org/10.1016/j.jclinepi.2014.04.006.

Birnbaum, Y., Kitakaze, M., Grieve, D., Uretsky, B.F., 2023. Who should be the corresponding author, what are their responsibilities, and what email address should they provide? Cardiovascular Drugs and Therapy 37, 1039–1040.

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.

Cohen, J., 1960. A coefficient of agreement for nominal scales. Educational and psychological measurement 20, 37–46.

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., Hermann, B., Lamprecht, A.L., Linxweiler, J., Prat, A., et al., 2025. Multi-dimensional research software categorization. Computing in Science & Engineering .

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.

Hirsch, J.E., 2005. An index to quantify an individual's scientific research output. Proceedings of the National academy of Sciences 102, 16569–16572.

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.

McHugh, M.L., 2012. Interrater reliability: the kappa statistic. Biochemia medica 22, 276–282.

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.
da Silva, J.T., Dobránszki, J., Van, P.T., Payne, W.A., 2013. Corresponding authors: rules, responsibilities and risks. Asian Australas J Plant Sci Biotechnol 7, 16–20.
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.
Teunis, T., Nota, S.P., Schwab, J.H., 2015. Do corresponding authors take responsibility for their work? a covert survey. Clinical Orthopaedics and Related Research® 473, 729–735.
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.
Weber, N.M., Thomer, A.K., 2014. Paratexts and documentary practices: Text mining authorship and acknowledgment from a bioinformatics corpus, in: Examining paratextual theory and its applications in digital culture. IGI Global, pp. 84–109.
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.

We enriched these article-repository pairs with metadata from multiple sources to create a comprehensive and analyzable dataset. 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% (n=78,021) of all DOIs within our dataset[12].

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 high 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 contain 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. We selected the three most similar authors for each developer account for annotation efficiency.

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 2,548 (~85.0%) negative matches, comprising 2,027 unique authors and 2,733 unique developer accounts.

Our collected data for annotation confirmed that exact matching would be insufficient—only 2,191 (~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 2,442 (~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:

[12]Broken out by dataset source, we resolved ~2.1% (n=125) of all PLOS DOIs, ~4.0% (n=93) of all JOSS DOIs, ~0.0% (n=0) of all SoftwareX DOIs, and ~49.2% (n=63,817) 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-5 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 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 2 presents a complete comparison of model architectures and feature combinations.

Table 2: 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 most significant 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 consistently performed well 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. Evaluation of Model Performance on Non-JOSS Author-Developer Pairs.

To assess our model's generalizability beyond the JOSS dataset used for training, we conducted an additional evaluation using author-developer-account pairs from the PLOS, SoftwareX, and Papers with Code datasets. We created a dataset of all combinations of possible author-developer pairs from 20 article-repository pairs from each dataset. This resulted in 535 possible author-developer pairs for annotation across the 60 total article-repository pairs. Two independent annotators classified each of the possible author-developer pairs as

either matches or non-matches. This annotation process mirrors how our complete dataset was constructed by using the trained model to predict matches from all possible author-developer pairs for a given article-repository pair. The two annotators achieved a Cohen's kappa of 0.948, or "almost perfect" agreement (Cohen, 1960; McHugh, 2012). Annotators further discussed and resolved all disagreements.

The trained author-developer matching model was then applied to this new dataset, achieving an overall binary-F1 score of 0.89, precision of 0.92 and recall of 0.87 (positive="match"). The overall macro-F1 score was 0.94, with a precision of 0.95 and recall of 0.93. Per-dataset performance is presented in Table 3. Across each of the non-JOSS datasets, the model demonstrated strong performance, with binary-F1 scores ranging from 0.88 to 0.90, and acheiving 0.94 for all macro-F1 scores.

Table 3: Per-dataset performance of the author-developer-account matching model on non-JOSS datasets.

dataset	binary-F1	binary-Precision	binary-Recall	macro-F1	macro-Precision	macro-Recall
PLOS	0.88	0.79	1.00	0.94	0.89	0.99
SoftwareX	0.90	0.96	0.85	0.94	0.97	0.92
Papers With Code	0.89	1.00	0.81	0.94	0.98	0.90

7.1.2.4. 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 do not correspond to individual authors. However, our filtering mechanisms applied before analysis help minimize their impact on modeling.

Additional limitations are discussed in a qualitative error analysis conducted as part of our evaluation of the code-contributing non-author sub-sample in @#sec-appendix-cc-na-error-analysis.

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:

- 53 days for preprints
- 114 days for research articles
- 282 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.

article_type	count	mean	std	min	10%	25%	50%	75%	90%	max
preprint	2683	110	182	-1520	0	6	53	138	285	2091

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.

article_type	count	mean	std	min	10%	25%	50%	75%	90%	max
research article	17017	193	253	-931	0	19	114	269	487	3176
software article	200	394	475	-1	0	50	282	536	951	3007

7.2. Distributions of Author-Developer-Account Prediction Confidence

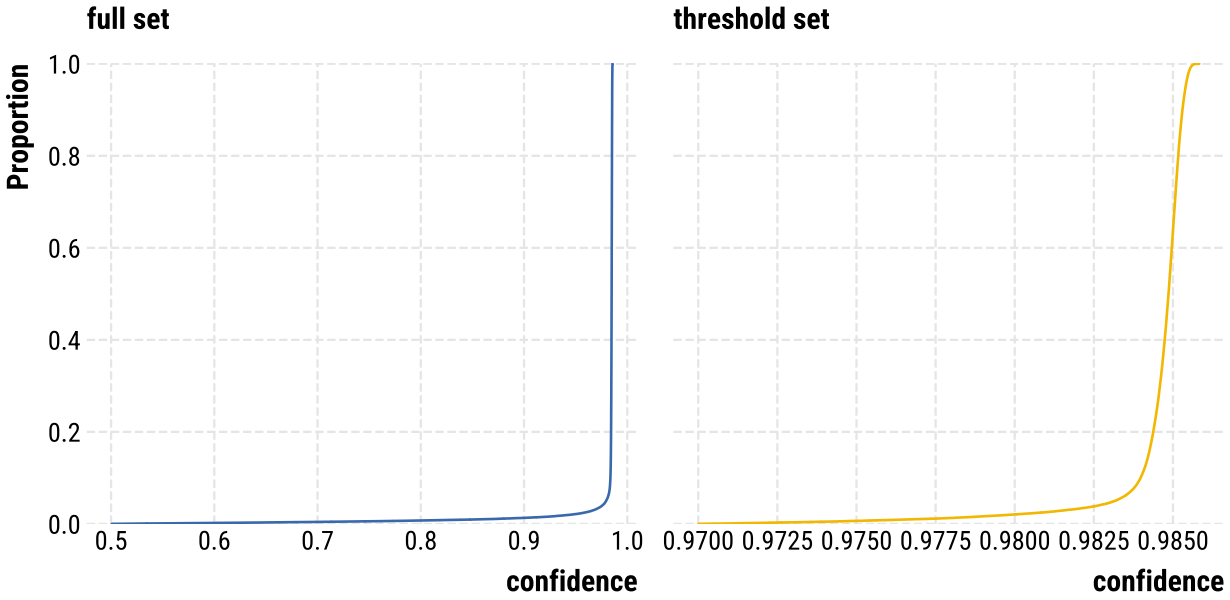

Figure 5: Distribution of author-developer-account prediction confidence scores. The plot on the left shows the distribution of all prediction confidence scores. The plot on the right shows the distribution of prediction confidence scores for author-developer-account pairs with a confidence score greater than or equal to 0.97.

Thresholding the predictive model confidence at 0.97 resulted in a ~3.2% (n=2,911) reduction in the number of author-developer-account pairs (from an unfiltered total of 90,086 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	931	4,927	1,188
	Life Sciences	1,679	7,877	2,252
	Physical Sciences	16,165	55,164	20,898
	Social Sciences	1,125	4,663	1,567
By Document Type	preprint	2,683	11,327	4,190
	research article	17,017	61,279	21,482
	software article	200	905	449
By Access Status	Closed	1,070	5,054	1,679
	Open	18,830	66,677	24,026
By Data Source	joss	77	353	246
	plos	2,109	10,345	2,729
	pwc	17,591	59,261	22,313
	softwarex	123	554	204
Total		**19,900**	**69,808**	**25,358**

7.3.1. Distributions of Team Composition

Table 6 provides detailed statistics on the composition of research teams across different domains, article types, and open access statuses.

Table 6: 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 repositories that have programming language files, 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 ± 1.8	4.0 ± 1.9	1.1 ± 0.7	0.6 ± 2.1
	Open	4.8 ± 1.8	3.8 ± 1.9	1.0 ± 0.7	0.5 ± 1.8
Domain	Health Sciences	5.8 ± 2.2	4.8 ± 2.3	1.0 ± 0.6	0.4 ± 1.3
	Life Sciences	5.0 ± 2.0	4.0 ± 2.1	1.0 ± 0.7	0.4 ± 1.2
	Physical Sciences	4.8 ± 1.7	3.7 ± 1.8	1.0 ± 0.7	0.5 ± 1.9
	Social Sciences	4.5 ± 1.7	3.4 ± 1.7	1.1 ± 0.7	0.3 ± 1.2
Article Type	preprint	4.8 ± 1.7	3.7 ± 1.8	1.0 ± 0.7	0.6 ± 2.3
	research article	4.8 ± 1.8	3.8 ± 1.9	1.0 ± 0.7	0.4 ± 1.7
	software article	4.6 ± 1.7	3.1 ± 1.8	1.5 ± 1.3	0.9 ± 1.1
Overall		**4.8 ± 1.8**	**3.8 ± 1.9**	**1.0 ± 0.7**	**0.5 ± 1.8**

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 code and documentation changes were labeled "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 near perfect agreement for contribution type (=0.89), and perfect agreement for author matching assessment and bot account detection (=1.0), each annotator independently labeled 90 contributors— 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 contributor to the article-repository pair.
- **Number of Additions**: The total number of lines of code added by the code contributor to the article-repository pair.
- **Number of Deletions**: The total number of lines of code deleted by the code contributor 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 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 amount of changes to the repository.

7.4.2. Results

We find that ~39% (n=78) of code-contributing non-authors were correctly classified as non-authors, ~30.5% (n=61) were unclear due to insufficient profile information, and ~30.5% (n=61) appeared to be missed classifications that should have been matched to authors. Only two 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 7 and Table 8 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 ~10.7% 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 comprised ~50.5% of total repository commits and ~41.7% of repository absolute changes, even at the 25th percentile.

Table 7: 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 8: 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 ~49.2% of unclear cases (n=30) owned the repositories they contributed to. This suggests that 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 9), the median contribution drops to ~34.6% of total commits and ~12.7% of absolute changes, though this still represents substantial technical involvement.

Table 9: 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.

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.

7.4.3. Qualitative Error Analysis of Missed Classifications

To better understand why certain code-contributing non-authors were missed classifications, we conducted a qualitative error analysis of the 61 contributors labeled as such. We identified several common themes:

1. **Limited Information from Text Alone**: The original dataset for model training and evaluation was constructed using only text-based features from author names and developer information. However, for this extended examination, annotators utilized the full code-contributor profile, including linked websites or linked ORCID profiles. This was done because we wanted to understand the nature of missed classifications (with more time and information to make a classification) rather than strictly replicating the model's text-only approach. From text alone, many of these missed classifications would have been very challenging to identify. This highlights a limitation in our current model, and a potential area for future work, such as incorporating details from linked websites or other contextual information to improve matching performance.
2. **Name Variations and Cultural Differences**: The model performed better with Anglosaxon names, while names from other cultures were more likely to be missed. This suggests possible bias in the training data and a clear area for future work.
3. **Additional Unrelated Text in Names**: When usernames or display names contained longer phrases or unrelated words, the model tended to classify them as no-match, even if there were strong indicators of a match. For example, a username such as "awesome_computational_biologist_john_d" paired with an author name "John Doe" might be missed due to the additional text in the username.
4. **Significant Differences Between Username and Author Name**: The model struggled when there were substantial differences between the username and author name, such as when an individual provides a chosen name in their GitHub profile that differed significantly from their authorship name. Most commonly this occurred when an individual used a chosen "English" name in their GitHub profile that was very different from their authorship name.

These themes highlight areas for potential improvement in the model, such as incorporating more diverse training data and exploring additional features that could capture cultural name variations and contextual information.

7.5. Article Citation Linear Model Results

Table 10: Article citations by code contributorship of the 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 ***	**0.95**	**0.00**	**0.90**	**1.01**
Total Authors ***	**0.08**	**0.00**	**0.07**	**0.09**
Code-Contrib. Authors ***	**0.04**	**0.00**	**0.02**	**0.06**
Code-Contrib. Non-Authors	-0.00	0.61	-0.01	0.01
Years Since Publication ***	**0.40**	**0.00**	**0.39**	**0.40**

Table 11: Article citations by code contributorship of the 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 ***	**0.65**	**0.00**	**0.52**	**0.78**
Total Authors ***	**0.08**	**0.00**	**0.07**	**0.09**
Code-Contrib. Authors	-0.03	0.48	-0.13	0.06
Code-Contrib. Non-Authors	0.01	0.44	-0.02	0.04
Years Since Publication ***	**0.39**	**0.00**	**0.38**	**0.39**
Is Open Access ***	**0.34**	**0.00**	**0.21**	**0.46**
Code-Contrib. Authors × Is Open Access	0.08	0.12	-0.02	0.17
Code-Contrib. Non-Authors × Is Open Access	-0.01	0.39	-0.05	0.02

Table 12: Article citations by code contributorship of the 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 ***	**0.87**	**0.00**	**0.73**	**1.02**
Total Authors ***	**0.08**	**0.00**	**0.07**	**0.09**
Code-Contrib. Authors	-0.00	0.96	-0.12	0.11
Code-Contrib. Non-Authors	0.02	0.50	-0.04	0.07
Years Since Publication ***	**0.40**	**0.00**	**0.39**	**0.41**
Domain Life Sciences **	**-0.22**	**0.01**	**-0.39**	**-0.06**
Domain Physical Sciences	0.11	0.12	-0.03	0.25
Domain Social Sciences **	**-0.25**	**0.01**	**-0.43**	**-0.07**
Code-Contrib. Authors × Domain Life Sciences	0.10	0.16	-0.04	0.23
Code-Contrib. Authors × Domain Physical Sciences	0.03	0.56	-0.08	0.15
Code-Contrib. Authors × Domain Social Sciences *	**0.14**	**0.05**	**0.00**	**0.29**
Code-Contrib. Non-Authors × Domain Life Sciences	-0.05	0.20	-0.12	0.02
Code-Contrib. Non-Authors × Domain Physical Sciences	-0.02	0.43	-0.08	0.03
Code-Contrib. Non-Authors × Domain Social Sciences	-0.03	0.36	-0.11	0.04

Table 13: Article citations by code contributorship of the 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 ***	**0.44**	**0.00**	**0.35**	**0.53**
Total Authors ***	**0.08**	**0.00**	**0.07**	**0.09**
Code-Contrib. Authors	-0.01	0.86	-0.06	0.05
Code-Contrib. Non-Authors **	**-0.03**	**0.00**	**-0.05**	**-0.01**
Years Since Publication ***	**0.41**	**0.00**	**0.40**	**0.42**
Article Type Research Article ***	**0.53**	**0.00**	**0.45**	**0.61**
Article Type Software Article **	**-0.46**	**0.00**	**-0.73**	**-0.19**
Code-Contrib. Authors × Article Type Research Article *	**0.07**	**0.04**	**0.00**	**0.13**
Code-Contrib. Authors × Article Type Software Article	-0.08	0.23	-0.22	0.05
Code-Contrib. Non-Authors × Article Type Research Article **	**0.04**	**0.00**	**0.02**	**0.06**
Code-Contrib. Non-Authors × Article Type Software Article	0.08	0.28	-0.07	0.24

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

Table 14: Counts of Code-Contributing Authors ('Coding') and 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	625	931	0.000***
		Middle	200	3508	0.000***
		Last	85	922	0.000***
	Life Sciences	First	1115	1678	0.000***
		Middle	384	5059	0.000***
		Last	214	1674	0.000***
	Physical Sciences	First	11318	16092	0.000***
		Middle	4458	44329	0.000***
		Last	1073	15956	0.000***
	Social Sciences	First	788	1122	0.000***
		Middle	358	2776	0.000***
		Last	120	1121	0.000***
Article Type	Preprint	First	1872	2662	0.000***
		Middle	791	7335	0.000***
		Last	174	2656	0.000***
	Research Article	First	11848	16961	0.000***
		Middle	4471	47814	0.000***
		Last	1276	16817	0.000***
	Software Article	First	126	200	0.002**
		Middle	138	523	0.000***
		Last	42	200	0.000***
Open Access Status	Closed Access	First	772	1066	0.000***
		Middle	298	3250	0.000***
		Last	70	1053	0.000***
	Open Access	First	13074	18757	0.000***
		Middle	5102	52422	0.000***
		Last	1422	18620	0.000***
Overall	Overall	First	13846	19823	0.000***
		Middle	5400	55672	0.000***
		Last	1492	19673	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).

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

Table 15: Counts of Code-Contributing Authors ('Coding') and 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	411	1974	0.000***
		Not Corresponding	499	3387	0.000***
	Life Sciences	Corresponding	867	3957	0.000***
		Not Corresponding	846	4454	0.000***
	Physical Sciences	Corresponding	2214	5975	0.000***
		Not Corresponding	14635	70402	0.000***
	Social Sciences	Corresponding	329	954	0.000***
		Not Corresponding	937	4065	0.000***
Article Type	**Preprint**	Corresponding	13	41	0.055
		Not Corresponding	2824	12612	0.000***
	Research Article	Corresponding	3739	12638	0.000***
		Not Corresponding	13856	68954	0.000***
	Software Article	Corresponding	69	181	0.003**
		Not Corresponding	237	742	0.000***
Open Access Status	**Closed Access**	Corresponding	82	181	0.468
		Not Corresponding	1058	5188	0.000***
	Open Access	Corresponding	3739	12679	0.000***
		Not Corresponding	15859	77120	0.000***
Overall	**Overall**	Corresponding	3821	12860	0.000***
		Not Corresponding	16917	82308	0.000***

Counts of authors in Table 15 may differ slightly from counts in Table 5. Table 5 counts unique authors, while Table 15 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 16: 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 17: 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 18: 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 19: 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 20: 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. Study Differences from Preregistration

7.10.1. Analysis of Article Field Weighted Citation Impact (FWCI) and Code Contribution

In our pre-registered analysis plan (https://osf.io/fc74m), we initially stated that we would additionally investigate the relationship between an article's Field Weighted Citation Impact (FWCI) and the number of code contributors to the project. We decided against this analysis as the FWCI metric was only available from OpenAlex for 55.5% (n=76904) articles from the `rs-graph-v1` dataset at the time of data processing. In addition, our analysis of the relationship between article citations and the number of code contributors to the project already includes the articles domain and duration since publication providing similar control.

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

In our pre-registered analysis plan (https://osf.io/fc74m), we initially hypothesized that there would be a positive 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 exclude this analysis 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 could benefit from a more granular analysis of individual contribution patterns, incorporating measures of contribution significance and sustainability rather than just temporal duration.