We Are Family: Analyzing Communication in GitHub Software Repositories and Their Forks

Scott Brisson
University of Toronto
Toronto, Canada
scott.brisson@mail.utoronto.ca

Ehsan Noei
University of Toronto
Toronto, Canada
e.noei@utoronto.ca

Kelly Lyons
University of Toronto
Toronto, Canada
kelly.lyons@utoronto.ca

***Abstract*—GitHub facilitates software development practices that encourage collaboration and communication. Part of GitHub's model includes forking, which enables users to make changes on a copy of the base repository. The process of forking opens avenues of communication between the users from the base repository and the users from the forked repositories. Since forking on GitHub is a common mechanism for initiating repositories, we are interested in how communication between a repository and its forks (forming a software *family*) relates to stars. In this paper, we study communications within 385 software families comprised of 13,431 software repositories. We find that the fork depth, the number of users who have contributed to multiple repositories in the same family, the number of followers from outside the family, familial pull requests, and reported issues share a statistically significant relationship with repository stars. Due to the importance of issues and pull requests, we identify and compare common topics in issues and pull requests from inside the repository (via branching) and within the family (via forking). Our results offer insights into the importance of communication within a software family, and how this leads to higher individual repository star counts.**

***Index Terms*—Empirical study, Open source software, Data mining, Software maintenance, Software family**

I. Introduction

GitHub is the largest open-source software host with almost one third of its 96 million repositories created in 2018 [1]. A unique feature to GitHub is its 'fork and pull model', which allows users to create copies of a repository and freely experiment without affecting the base repository. Forks are also used for requesting changes to the base repository, or act as starting points for different ideas [2]. As such, forking a repository creates an opportunity for social relationships to form between users in the base repository and users in the forked repositories. It has been shown that studying a repository's forks is important to account for all activity within a software project [3], [4].

Furthermore, forks are one of many factors correlated with stars, including number of committers, issues, and new contributions by popular users [5]–[7]. However, many studies have not investigated how users who are creating the forks, issues, and contributions participate in related sets of repositories such as within a repository and its forks. Our study deconstructs several communication metrics to determine how users' involvement relative to different repositories relate to star count.

We define a software **family** as a repository (that is not forked from another) and its forks. We refer to the base repository as the **parent** repository, and its forks as its **children**. Each child can also be forked into another repository, thus forming a family of repositories capable of creating social relationships through direct communication lines in software artifacts such as pull requests (PRs), issues, and mentions. A family f_i can be more formally defined as:

$$f_i = \{r_j^i\}, j = 1, ..., m_i \tag{1}$$

In equation (1), r_1^i is the parent repository, $r_j^i, j \neq 1$ are the children, and m_i is the number of repositories in family i.

GitHub offers users the ability to *star* a repository. A star can be used by users to show appreciation for a repository, keep track of its activities, and discover related content on their main feed [8]. Many studies have used stars to represent popularity [7], [9]. Popularity is useful for open source communities, as more popular repositories attract new contributors and relay to the repository's developers that their software is being used [9]. Studying factors that are associated with stars is useful for maximizing repository popularity.

In this paper, we analyze the importance of communication within active software families in relation to stars. We address the following questions:

RQ1) How does repository star count vary within and among families? We are interested in determining which repositories in a family have the most stars. Users from child repositories can create PRs and issues for their parent repositories and vice versa, possibly leading to more engaged repositories and higher star count. We conducted a Kruskal-Wallis test to compare the distribution of stars among families, which showed that some family star distributions differ [10]. A post-hoc Dunn's test revealed that 76% of the families we analyzed have similar star distributions, with the parent having the majority of stars [11]. We discovered it is rare for child repositories to surpass the star count of their parents, and stars decrease as repositories are forked further away from the parent.

RQ2) How much communication takes place among repositories within a family? We are interested in determining the amount of communication that takes place within and among families. In particular, we are interested in determining

978-1-7281-5143-4/20 © 2020 IEEE

SANER 2020, London, ON, Canada
Research Papers

if there is significant communication between users in the same family. We calculated several metrics of communication between users within a repository, within the family, and outside the family. We observed that the majority of PRs are with family members, many users contribute to repositories across the same family, and the fewest number of issues are reported by users in the same family. We conclude that there is a significant amount of communication that takes place between repositories in the same family, even though the majority of communication takes place between users from the same repository, and between users from the same repository and outside the family.

RQ3) How does communication, in the context of a family, relate to stars? Since **RQ2** has shown that there is a significant amount of communication that takes place between repositories in the same family, we are interested in determining the significance of familial communication on repository star count. We conducted a linear regression analysis and found that the fork depth, the number of users that have contributed to repositories in the same family, the number of followers from outside the family, and PRs between repositories in the same family have a significant relationship with star count. We then conducted a relative importance analysis of communications among users within repositories, among users within the family, and among users outside the family. We found that communication within families has the strongest relationship with star count.

RQ4) What are the topics within the communications? In **RQ3**, we found that the number of reported issues and PRs among repositories in the same family have a significant positive relationship with stars. Therefore, we are interested in whether issues reported from users in the same repository, family, or outside the family communicate about different topics. We are also interested if PRs from within the repository or PRs with family members discuss different topics. We conducted structural topic modeling on 59,023 issues and 65,756 PRs. With respect to issues, we determined that, depending on whether users are from the same repository, from the same family, or from outside the family, they report issues on different topics. However, with respect to PRs, there was no observable difference in topics from inside the repository or from the family.

Our contributions are as follows: **(a)** we introduced the notion of a software **family** to differentiate interactions from different users relative to each repository, **(b)** we discovered that communications between users in a repository and users in its family share a significant relationship with stars, and **(c)** we showed that users report issues on different topics depending on whether they are from the repository, family, or outside the family. The findings of this study emphasize the importance of collaborations between users working on related code-bases.

II. Study Setup

A. Data Collection

GHTorrent provides a mirror to GitHub's REST API and repository history, allowing access to terabytes of information hosted by GitHub [12]. The GHTorrent project continuously gathers data from the GitHub public event timeline, and stores them in a publicly available MySQL database. We used the `mysql-2019-03-01` dataset, which holds data on 116,217,069 repositories and 30,600,309 different users.

Starting with 116,217,069 repositories, we performed the following 7 steps to identify software families for our dataset:

1) Removing Non-collaborative Repositories: We are interested in repositories that use the collaborative aspects of GitHub. We removed repositories that have not participated in PRs or GitHub's issue tracker, which narrowed our dataset to 16,382,096 repositories.

2) Family Identification: We classified families by first identifying parents within the dataset. This was done by choosing repositories that were not forked from another. If a repository was not forked, it was labelled as a parent. We then recursively iterated through each parent's children until an entire family was formed. This resulted in 7,764,074 families composed of 16,250,265 repositories. We excluded 141,831 repositories whose parent repository was deleted. When a parent repository is deleted, GitHub selects one of its children as the new parent repository [13]. Even though subsequent children can be created from this new parent, GHTorrent does not update its information about the new parent. Therefore, we were unable to identify the parent in families whose original parent was deleted.

3) Removing Small Families: A minimum of 10 members per family is required for stable results in a logistic regression analysis [14]. Therefore, we removed families with fewer than 10 repositories. This reduced our dataset to 143,945 families with 5,589,713 repositories.

4) Removing Personal Repositories: Despite GitHub's commitment to social coding, many repositories on GitHub are personal repositories [3]. Because we are interested in studying communication among multiple users, we removed families with only 1 unique committer, resulting in a dataset of 143,566 families composed of 5,582,280 repositories.

5) Selecting Software Repositories: A large portion of repositories on GitHub are not used for software development [3]. To obtain a representative sample of software repositories with 95% confidence, we randomly sampled with replacement 385 parent repositories in our dataset. We categorized each repository as 'software' or 'other' using the classification system from Kalliamvakou *et al.* [3]. We kept each parent repository classified as 'software' for our dataset, and excluded each parent repository classified as 'other', until 385 parents were selected. Repositories that could not be categorized due to a language barrier were not used in our analysis; only English repositories were selected. Our classifications were verified with an independent source, a computer science graduate student with industrial and GitHub experience, with 93% agreement.

Our final dataset F is formed of 385 software families comprised of 13,431 repositories:

$$F = \{f_i\}, i = 1, ..., 385 \quad (2)$$

6) Mining Discussion Comments: GHTorrent provides a `comment_id` for comments in PRs and issues. However, it does not contain the comments themselves. We mined GitHub using GitHub's API for PR and issue comments for each repository. We then matched the mined comment IDs to the comment IDs found in GHTorrent to stay consistent with GHTorrent's data.

7) Identifying Repository Users: Lastly, as of 2014, GitHub disabled the API endpoint used to retrieve user repository memberships. We used the heuristics described by GHTorrent to create a list of users belonging to each repository by looking at original commit authorship and mergers of PRs [13]. Fake users as defined by GHTorrent were excluded. The set of users for each repository r_j^i in family f_i is:

$$U_j^i = \{u_k^j\}, k = 1, ..., n_j \tag{3}$$

In equation (3), i is the family number, j is the repository number, and n_j is the number of users belonging to to r_j^i.

B. Data Pre-processing

We performed the following 3 steps to calculate communicative data for each software family:

1) Sentiment Analysis: We first identified the language of each comment obtained in our data collection phase. `langid.py` is an off-the-shelf language identification library pre-trained on 97 languages with high accuracy across all domains [15]. We determined the language of each comment as the highest probable language reported by `langid.py`. We manually inspected a sample of 384 comments and found `langid.py` to be 95.6% accurate. We then determined the sentiment of each English comment using SentiStrength-SE, a domain-specific sentiment analysis tool for software engineering texts [16]. Because SentiStrength-SE is trained on English comments, non-English comments were excluded from our sentiment analysis. SentiStrength-SE reported two sentiment strengths for each comment: a positive score and a negative score. The positive score returns a 1 (not positive) to 5 (extremely positive), and the negative score returns -1 (not negative) to -5 (extremely negative). Two scores are returned separately because people process negative and positive emotion in parallel, as shown by Thelwall *et al.* [16]–[18]. Because of this, we also treat them as separate metrics in our analysis. Table I shows examples of SentiStrength-SE's output.

2) Calculating Star Count: Stars are important for open-source software repositories on GitHub, as more stars attract new developers, and indicate to developers that their software is being used [9]. Stars are used to show appreciation and interest for a repository. Many studies have used stars as a representation for popularity [7], [9].

Forks have also been used as a proxy for repository popularity and are highly correlated with stars [19]–[21]. Commits, programming language, and application domain have also been shown to be correlated with stars [9], [22]. However, due to the importance of stars on GitHub, and because our study is focused on families which are defined by forks, we chose to use stars as our response variable. The number of stars per repository is readily available in our dataset.

TABLE I: SentiStrength-SE examples.

Comment ID	Positive Score	Negative Score	Comment
223111194	4	-1	**Thank you!** This works like a charm! And also thank you very much for providing this really awesome theme!
333778837	1	-5	Same here... Makes me very sad :(
55693166	2	-2	Other than the poor config name, looks good

3) Calculating Explanatory Metrics: Table II shows a complete list of 66 metrics calculated for our linear regression model. We selected communication metrics and measured them relative to each repository as follows: a) users within the repository (ending in $\_repo$), b) users in the family (ending in $\_family$), and c) users not in the family (ending in $\_outside$). For a given metric category (e.g. issues), each user belongs exclusively to one of these categories. For example, the repository r_1^4 of family f_4 (containing l repositories) has 378 issues, with 68 issues reported by users U_1^4, who are members of r_1^4, 41 issues reported by users $U_1^4 \cap \{U_2^4 + U_3^4 + ... + U_l^4\} = \{\}$, who are members of f_4 (and not r_1^4), and 269 issues reported by users $\{U_1^4 + U_2^4 ... + U_l^4\}'$, who are not members of f_4. In Table II, these metrics are defined as $issues\_repo$, $issues\_family$, and $issues\_outside$, respectively. If an issue was reported by a user belonging to both r_1^4 and a different repository in the same family f_4, this issue will count towards category a) users within the repository. If an issue was reported by a user belonging to both f_4 and a different repository outside the family, this issue will count towards category b) users within the repository. If an issue reported by a user who belongs to neither r_1^4 or a different repository in the same family f_4, this issue will count towards category c) users not in the family. Our metrics fall under the following categories:

Forks. We calculated the number of $forks$ per repository, and the number of $active\_forks$ (forks that have participated in an issue or PR within the family) per repository, to validate how the number of forks relate to stars. We also calculated $depth$, the amount of subsequent forking from existing forks, as coined by Lima *et al.* [23]. Figure 1 follows the tree-like structure also noted by Lima *et al.* [23], with the deepest repository of f_{264} having $depth = 3$, and the deepest repositories of f_{62} having $depth = 1$. Parents of both families have $depth = 0$. This is to measure the relationship between star count and distance from the parent.

Users. Previous research has shown that the number of contributors is weakly correlated with stars [9]. We considered the number of unique $users$ U_j^i per repository, as well as the number of unique users per repository r_j^i that have also contributed to another repository in the same family, $U_j^i \cap \{U_1^i + U_2^i + ... + U_m^i\}$, to determine their respective relationship to stars.

TABLE II: Metrics calculated for each repository in each family.

Category	Metric	Description
Response	stars	Number of stars.
Age	age	How long (in hours) the repository has existed.
Forks	depth	Number of forks away from the parent repository.
	forks	Number of forks created from the repository.
	active_forks	Number of forks created from the repository that opened a PR or issue within the family.
Users	users_repo	Number of users that have write access to the repository.
	users_also_in_family	Number of users that have write access to the repository, and to at least one other repository in the same family.
Followers	followers_repo	Total followers of each user in the same repository.
	followers_family	Total followers of each user in the same family.
	followers_outside	Total followers of each user outside the family.
Pull Requests	pr_repo	Number of PRs that have occurred within the repository (via branching).
	pr_family	Number of PRs that have occurred within the family (via forking).
PR Comments	pr_repo_comments	Number of discussion comments from a PR within the repository.
	pr_family_comments	Number of discussion comments from a PR within the family.
	pr_repo_code_comments	Number of code comments from a PR within the repository.
	pr_family_code_comments	Number of code comments from a PR within the family.
PR Sentiments †	positive_sentiment	The average positive sentiment calculated for each metric in the `PR Comments` category.
	negative_sentiment	The average negative sentiment calculated for each metric in the `PR Comments` category.
PR Mentions	pr_mentioned_repo	Number of users mentioned in a PR belonging to the same repository.
	pr_mentioned_family	Number of users mentioned in a PR belonging to the same family.
	pr_mentioned_outside	Number of users mentioned in a PR from outside the family.
	pr_mentioned_unique_repo	Number of unique users mentioned in a PR belonging to the same repository.
	pr_mentioned_unique_family	Number of unique users mentioned in a PR belonging to the same family.
	pr_mentioned_unique_outside	Number of unique users mentioned in a PR from outside the family.
Issues	issue_repo	Number of issues opened by a user belonging to the same repository.
	issue_family	Number of issues opened by a user belonging to the same family.
	issue_outside	Number of issues opened by a user from outside the family.
Issue Comments	issue_comments_repo	Number of comments from issues opened by a user belonging to the same repository.
	issue_comments_family	Number of comments from issues opened by a user belonging to the same family.
	issue_comments_outside	Number of comments from issues opened by a user from outside the family.
Issue Sentiments ∗	positive_sentiment	The average positive sentiment calculated for each metric in the `Issue Comments` category.
	negative_sentiment	The average negative sentiment calculated for each metric in the `Issue Comments` category.
Issue Mentions	issue_mentioned_repo	Number of mentions in an issue opened by a user from the repository.
	issue_mentioned_family	Number of mentions in an issue opened by a user from the family.
	issue_mentioned_outside	Number of mentions in an issue opened by a user outside the family.
	issue_mentioned_unique_repo	Number of unique mentions in an issue opened by a user from the repository.
	issue_mentioned_unique_family	Number of unique mentions in an issue opened by a user from the family.
	issue_mentioned_unique_outside	Number of unique mentiona in an issue opened by a user outside the family.
Issue Events ∗	issue_closed	Number of issues closed.
	issue_subscribed	Number of users who have subscribed to an issue.
	issue_unsubscribed	Number of users who have unsubscribed from an issue.
	issue_reopened	Number of users who have reopened an issue.
	issue_assigned	Number of users assigned to an issue.
	issue_referenced	Number of users referencing an issue.

∗ All metrics have been calculated for users in the same repository, family, and outside the family.
† All metrics have been calculated for users in the same repository and family.

Followers. We considered the set of users U_j^i for each repository r_j^i. We calculated the total number of followers limited to U_j^i, the total number of followers from the same family f_i, and the total number of followers from outside f_i to determine the importance of familial following behavior to repository stars.

Pull Requests. We calculated the number of PRs within each repository r_j^i, as well as each PR with other repositories in the same family f_i. This is to compare the relationship of PRs with the family and PRs in the repository with star count.

PR Comments. We considered the number of PR comments coming from users within the same repository U_j^i, as well as comments from users belonging to repositories in the same family $U_j^i \cap \{U_1^i + U_2^i + ... + U_m^i\} = \{\}$ in order to determine the relationship between comments and stars.

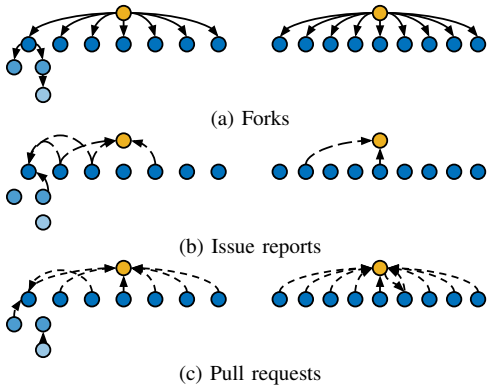
PR Sentiments. A positive or negative sentiment may affect willingness to star a repository. We performed a sentiment analysis for all PR comments in the `PR Comments` category.

PR Mentions. Mentioning a user in a `PR Comment` who is from the same repository, family, or outside the family may be representative of more social engagement, which may lead to more stars. We calculated the number of @mentions for each user in the repository, in the family, and outside the family.

Issues. Previous research has shown that the number of issues in a repository is positively correlated with forks [6], which is correlated with stars. We measured how many users from the repository, family, or outside the family are opening issues to determine each respective effect on stars.

Issue Mentions. Similar to `PR Mentions`, we calculated the number of @mentions in issue comments for each user in the repository, in the family, and outside the family.

Issue Comments. We considered the number of issue comments from users within the same repository, users belonging to repositories in the same family, and users outside the family.



(a) Forks

(b) Issue reports

(c) Pull requests

Fig. 1: Software families f_{264} (left) and f_{62} (right) with $|f_{264}| = 11$ and $|f_{62}| = 10$, showing their a) forks, b) issues, and c) PRs between family members (volume omitted).

Issue Sentiments. Similar to `PR Sentiments`, we performed a sentiment analysis for all comments in the `Issue Comments` metric.

Issue Events. Certain actions can be performed on issues such as closing, subscribing, assigning, and referencing. Using these different GitHub events may be reflective of a more active repository, which may affect the star count. We considered the number of events triggered by users in the same repository, within the family, and outside the family.

III. Approach and Results

RQ1) How does repository star count vary within and among families?

Motivation. We are interested in the distribution of stars among families to compare parent repository star count with their children's. Users in child repositories can create PRs and issues for their parent repositories and vice versa, possibly leading to more engaged repositories and more stars.

Approach. We tested our families for homogeneous variances using a Levene test [24]. We then conducted a Kruskal-Wallis test to compare the star distribution among our 385 collected families [25]. We performed a post-hoc Dunn's test to determine differences in family star distributions, then compared the distribution of stars between parent repositories and their children [26].

Findings. The Levene test showed that our families have homogeneous variances (p = 1), which allowed us to conduct a Kruskal-Wallis test. Our Kruskal-Wallis test showed that at least one of our families did not have the same distribution (p = 7.537e-08). We then conducted a post-hoc Dunn's test, which revealed differing family distributions. We separated families with similar distributions, and found that 293 families (76%) had similar distributions.

We then binned the total number of stars by fork depth. Fig. 2 shows that most stars from our sample belong to the parent repositories, with stars decreasing the further children are from the parent. This is despite the fact that most children in our sample are immediate forks from the parent.

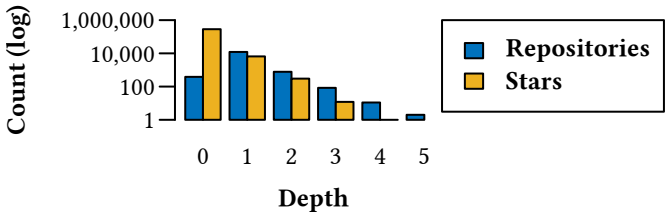


Fig. 2: Number of repositories and stars at different depths.

Lastly, we compared the star count of each parent in relation to their children, and found that only 9 parents (2.3% of our sample) have fewer stars than their children, with 11 repositories at $depth = 1$ and 3 repositories at $depth = 2$ having more stars than their parents.

76% of the families follow the same distribution of stars, where parent repositories have received the majority of stars, and children rarely surpass their parents' star counts.

RQ2) How much communication takes place among repositories within a family?

Motivation. Table II is a complete list of metrics calculated for our model, with most calculated relative to each repository, family, and outside the family. We are interested in exploring where the communication is coming from, and in what quantities, to determine which modes of communication are used more or less by users from the same repository, from the family, and from outside the family.

Approach. We calculated our metrics using the data found in the GHTorrent data set, and compared the volume of each metric relative to the repository, family, and outside the family to determine more or less frequent modes of communication.

Findings. Figure 3 shows the count of all metrics in our sample. For example, the `PRs` metric in figure 3 shows that PRs occur more often with the family than with the repository. Our findings as they relate to software families are as follows:

1) We compared the number of PRs with the family, and within the repository from our sample of 65,756 PRs. Of these PRs, 40.4% (26,590) were with branches, and 59.6% (39,166) PRs were with family members.
2) There are 24,998 unique users in our sample. Out of all users, 11,499 have contributed to multiple repositories in the same family.
3) From our sample consisting of 59,023 issues, approximately 29.6% of issues are reported from users in the repository, 9.7% from users in the family, and 60.1% from users outside the family.

The majority of PRs happen within the family, and many users contribute to repositories across the same software family. The majority of issues are opened by users outside the family, with the fewest opened by users from the family.

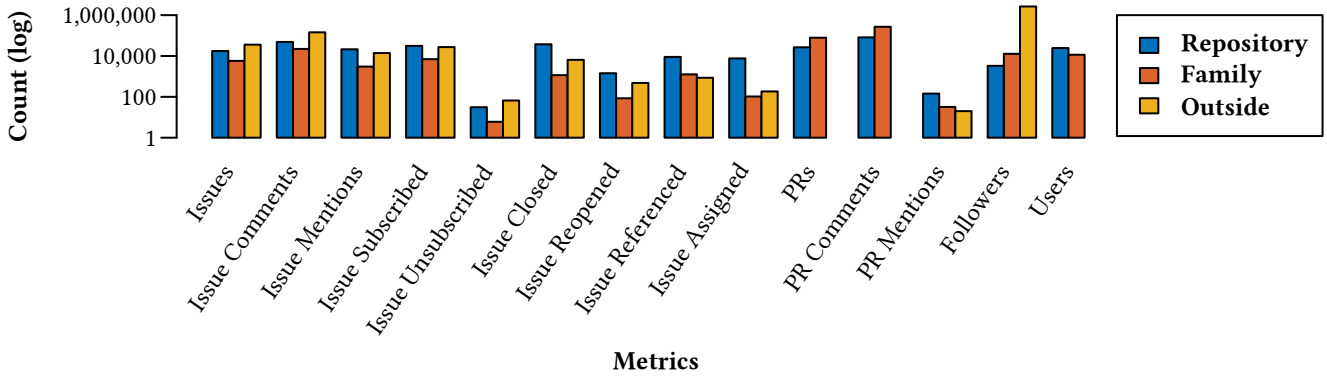


Fig. 3: Metric counts separated by repository, family, and outside the family.

While most communications are from users in the same repository or from outside the family, Figure 3 shows there is still a significant amount of communication between repositories in the same family. We also made more interesting observations related to communication as follows:

1) Few repositories in our sample use GitHub's issue tracker. Of our sample, only 2,595 repositories use GitHub's issue tracker. This observation aligns with existing research [6]. Interestingly, each parent of our 385 families has at least one registered issue.
2) We observe more discussion in issues than in PRs. Our sample contains 59,023 issues with 215,489 comments, compared to 65,756 PRs with 139,501 PR comments, averaging to 3.65 and 2.12 comments respectively.
3) Our sample only contains 196 mentions from PRs, compared with 38,182 mentions within our sample's issues. While our sample has more comments from issues (215,489) compared with PRs (139,501), the proportion of mentions between the two are are very different. These results support previous research, which shows that mentions are uncommon among PRs [27].
4) Very few users unsubscribe from issues. From our sample, there are 65,396 instances of users subscribing to events, with only 103 instances of users unsubscribing.

RQ3) How does communication, in the context of a family, relate to stars?

Motivation. As shown in **RQ2**, there is an abundance of interactions within a software family. We investigated whether interactions between family repositories are related to stars. We compared these familial interactions with interactions in the repository itself, with users outside of the family, and their relationship to repository star count.

Approach. We implemented a linear regression model using a subset of metrics from Table II, with the number of stars in each repository as the response variable [25]. We prepared our model by conducting a hierarchical cluster analysis to determine correlated metrics with Spearman's $|\rho| > 0.7$ [28], [29]. Including correlated variables in linear regression models negatively affects the stability of linear models, and hides the impact of each metric on the response variable [30]. Table III shows all correlated metrics that were removed from our model (right column). Each representative metric (left column) was included in our model, and chosen based on its simplicity to calculate and interpret. Lastly, we normalized the predictor variables to properly compare coefficients [29].

TABLE III: Representative and correlated metrics.

Representative Metric	Correlated Metrics
users_also_in_family	users_repo
pr_mentioned_repo	pr_mentioned_unique_repo
pr_mentioned_family	pr_mentioned_unique_family
pr_mentioned_outside	pr_mentioned_unique_outside
pr_family_code_comments	pr_family_code_pos
pr_repo_code_comments	pr_repo_code_pos pr_repo_code_neg
issue_outside	issue_mentioned_repo issue_mentioned_unique_repo issue_subscribed_repo issue_family issue_comments_family issue_family_pos issue_family_neg issue_mentioned_family issue_mentioned_unique_family issue_subscribed_family issue_comments_outside issue_outside_pos issue_outside_neg issue_closed_outside issue_mentioned_outside issue_mentioned_unique_outside issue_subscribed_outside
forks	active_forks
issue_repo	issue_closed_repo issue_comments_repo issue_referenced_repo issue_repo_pos issue_repo_neg

Findings. Table IV shows the results of our linear regression analysis, with $R^2 = 0.4183$. Our findings as they relate to software families are as follows:

1) We find that the *depth* of a repository in a family has a significant negative relationship with stars, as observed in **RQ1**.
2) Interestingly, we also find that the number of users who have contributed to other repositories in the *same family* has a positive relationship with stars. Previous

TABLE IV: Results from linear regression model.

The **Metric** column refers to each metric included in our linear regression model. The **Estimate** column is the intercept, **Pr(>F)** is the p−value, and **Rel.** is the relationship between the metric and star count.

Metric	Estimate	Pr(>F)		Rel.
Non-communicative	–	–		–
forks	14050.658	< 2e-16	***	↗
depth	277.189	2.40e-14	***	↘
age	12.583	0.375914		
Repository	–	–		–
issue_repo	4725.571	< 2e-16	***	↗
issues_unsubscribed_repo	5693.965	< 2e-16	***	↘
pr_mentioned_repo	5223.013	5.95e-15	***	↘
issues_assigned_repo	3460.600	3.41e-14	***	↘
followers_repo	1539.621	2.24e-12	***	↘
pr_repo_code_comments	1685.611	1.22e-08	***	↗
pr_repo_neg	231.079	0.008786	**	↘
pr_repo_comments	4086.671	0.008076	**	↘
pr_repo	2386.168	0.026081	*	↘
pr_repo_pos	40.304	0.613545		
issues_reopened_repo	1399.081	0.238119		
Family	–	–		–
users_also_in_family	1971.051	< 2e-16	***	↗
pr_mentioned_family	5184.101	< 2e-16	***	↗
issues_unsubscribed_family	6435.359	< 2e-16	***	↗
issues_closed_family	7114.712	< 2e-16	***	↘
issues_reopened_family	5383.633	< 2e-16	***	↗
issues_assigned_family	2335.077	< 2e-16	***	↗
issues_referenced_family	2089.421	3.11e-07	***	↘
pr_family_code_comments	1219.366	0.000627	***	↘
pr_family	849.382	0.003637	**	↗
pr_family_comments	747.066	0.008731	**	↘
followers_family	130.352	0.018029	*	↘
pr_family_pos	27.776	0.228881		
pr_family_code_neg	11.466	0.776198		
pr_family_neg	3.041	0.899862		
Outside	–	–		–
followers_outside	2441.401	< 2e-16	***	↗
issues_reopened_outside	9066.449	< 2e-16	***	↗
issue_outside	6506.352	5.13e-09	***	↗
issues_unsubscribed_outside	1703.198	7.90e-08	***	↗
pr_mentioned_outside	1056.548	4.88e-06	***	↘
issues_referenced_outside	1380.460	0.013112	*	↘
issues_assigned_outside	170.946	0.751762		

research has shown that the number of contributors in a single repository ($users\_repo$) is correlated with stars [5]. In our study, $users\_repo$ is correlated with the number of users who have also contributed to other repositories ($users\_also\_in\_family$). Including $users\_also\_in\_family$ in our model resulted in a better goodness-of-fit.

3) We observed that the total number of followers of users belonging to the repository and belonging to the family has a negative relationship with stars. However, we find that repositories with members who have more followers from outside the repository have a significant positive relationship with stars. This may support previous studies which show that followers are likely to star new repositories after a user they are following contributes to that repository [7], [31].
4) PRs can be done via forking and branching. We found that more PRs with family members (pr_family) has a statistically significant relationship with stars, whereas repositories that rely more on PRs from inside the repository (pr_repo) has a negative relationship with stars. This may be due to more engagement with different users outside the repository, but future studies are required to see if this is the case.

The number of users who contribute to other repositories in the same family share a significant positive relationship with stars. The number of followers from outside the family has a significant positive relationship with stars, whereas the number of followers from inside the repository and family is negatively associated with stars. The number of PRs with the family has a statistically significant positive relationship with stars, whereas PRs from inside the repository has a negative relationship with stars.

We also made more observations related to stars as follows:

1) We find that age does not have statistically significant relationship with stars, which corroborates previous findings [9]. This is likely because different repositories gain stars at different rates [21], [32].
2) We also find that the number of forks also has a significant positive relationship with the number of stars. This is in line with the findings of other research results [9].
3) We find that PR sentiment, from within the repository and family, does not share a statistically significant relationship with stars, while the average negative sentiment of PRs within the repository (pr_repo_neg) and the number of PRs from within the repository (pr_repo) has a negative relationship with star count. Therefore, we cannot with certainty conclude that the average negative sentiment of PRs within the repository has a statistically significant relationship with stars.
4) We find that the number of issues, no matter if they are opened from users inside the repository, the family, and outside the family, has a significant relationship with stars. Previous research has shown that the number of reported issues is strongly correlated with the number of forks and watchers, which validates these results [6].
5) We find that 5 issue events ($issue\_unsubscribed\_outside$, $issue\_reopened\_outside$, $issue\_assigned\_family$, $issue\_reopened\_family$, and $issue\_unsubscribed\_family$) have a positive relationship with stars, and 5 issue events ($issue\_referenced\_outside$, $issue\_referenced\_family$, $issue\_closed\_family$, $issue\_assigned\_repo$, and $issue\_unsubscribed\_repo$) have a negative relationship with stars.

To determine the importance of families within the context of individual repository star count, we performed relative importance testing with our metrics to determine which metrics had the most effect on our model's fit. We grouped each metric into four categories depending on their characteristics:

- *Non-communicative*. Metrics relating to the repository itself (i.e. age, $depth$ and $forks$).

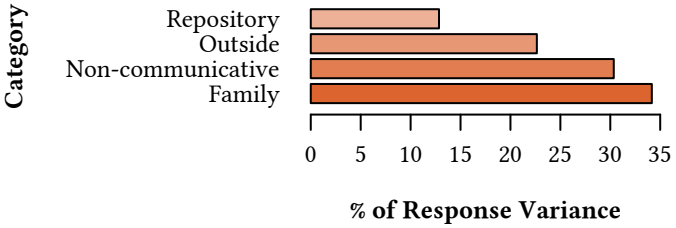


Fig. 4: Relative importance (pmvd) score.

- *Repository*. Metrics involving communication with only users from the same repository.
- *Family*. Metrics involving communication with users from the same repository's family.
- *Outside*. Metrics involving communication with users from outside the repository's family.

Specific groupings of each metric can be found in Table IV. We compared each grouping using proportional marginal variance decomposition (`pmvd`) via the `relaimpo` package [33].

Figure 4 shows the contribution of each grouping to the fit of our model. Our results show that our *family* communication metrics contributed the most to the fit of our model at 34.1%, followed by the repository's *non-communicative* metrics at 30.3%, *outside* communication metrics at 22.7%, and lastly *repository* communication metrics at 12.8%.

Interactions involving family members contribute the most to repository star count.

RQ4) What are the topics within the communications?

Motivation. In **RQ3**, we observed that regardless of who reports an issue, the number of issues in relation to stars is significant. Due to the significance of these findings, we determined the topics within these issues to investigate whether certain topics were discussed more often inside the repository, the family, or outside the family. We also observed that PRs within a repository have a negative relationship with stars, and PRs with family members have a positive relationship with stars. We also investigated topics for both familial and repository PRs to see how topics within these PRs differ.

Approach. We performed structural topic modeling on comments from 57,097 issues and 66,504 PRs using the `stm` and `tidytext` packages [34], [35]. Each comment was pre-processed to remove stop words, numbers, emails, and URLs. The number of topics was calculated using Deveaud *et al.*'s method by maximizing information divergence between different topics via the `ldatuning` R package [36], [37]. We calculated 20 topics for our issue set, and 14 topics for our PR set. We used `stm`, the R package for structural topic models to determine the topics of our issues [34]. The configuration for our model was calculated automatically via a moment-based estimator [38]. 55,667 issues and 62,952 PRs were successfully classified, with 1,430 issues and 3,552 PRs were dropped as they no longer contained any text after pre-processing. We then manually named and described each topic by looking at the top 5 word probabilities and a random sample of 10 issues and PRs. The topic names and descriptions were manually assigned then shared with a graduate student in computer science (with industrial and GitHub experience). After discussion and iteration, a final set of names and descriptions were agreed upon.

Findings. Table V shows the name, description, and top 5 words for each topic calculated from our issues. We know from **RQ2** that approximately 29.6% of issues are reported from users in the repository, 9.7% from users in the family, and 60.1% from users outside the family. Figure 5 shows the breakdown of each topic.

There are several topics that are application specific. T_1 `NextCloud Config`, T_{10} `Nextcloud`, and T_{16} `Ruby` are application specific, with 1, 4, and 35 families out of 354 families discussing issues with these topics with a minimum of 50% certainty. It is interesting to note that issues with these topics are opened by users who are mainly from outside the repository (89.4% for T_1, 75.6% for T_{10}, and 73.3% for T_{16}), indicating that for these applications, issues are driven mainly from outside the family.

However, there are certain issue topics discussed by many families. For example, T_6 `Features` is discussed by 310 families out of 354 families with a minimum of 50% certainty, and are more often reported by users from the same repository at 45.7%, followed by 42.4% of users outside of the family. T_3 `Styling` is also discussed among many families, with 172 of our 354 families discussing this topic with a minimum of 50% certainty. The distribution of users reporting issues discussing these topics aligns closely with the findings from **RQ2** with 30.1% issues reported from users within the repository, 10.4% issues reported from users from the family, and 59.5% issues reported from users outside the family.

There are also certain topics reported more often than average by users from the family. Issues discussing T_{18} `Builds`, are reported more often than average by users in the same family at 18.5%. Issues discussing T_{13} `Html and Latex` have been reported by users of the same family at 14.7%. Lastly, T_9 `Logs` is also discussed more often than average in issues opened by users in the family at 13.1%.

Table VI shows the name, description, and top 5 words for each topic calculated from our PRs. We know from **RQ2** that approximately 40.4% of PRs are with branches, and 59.6% of PRs are with the family. Figure 6 shows the breakdown of each PR topic depending if they came from users in the repository, or users from the family.

Topics P_1 `NextCloud Config.`, P_3 `Styling`, P_4 `Vr and HAS`, P_6 `Features`, and P_{11} `Node` are discussed mostly from PRs with the family. These topics are also extremely application specific, with only 23, 28, 17, 26, and 20 families out of 384 families having a PR identified with these topics with a minimum of 50% certainty. Our topic modeling also grouped PRs based on PR enhancement applications. P_5 `Methods`, P_6 `Features`, P_8 `Presto`, P_9 `Logs`, and P_{12} `Build and Installation` are all PR topics using

TABLE V: Descriptions for each issue topic.

Topic	Topic Name	Top Word Probabilities	Brief Description
T_1	NextCloud Config.	nextcloud, details, summary, server, removed	Server configuration issues, mainly within the Nextcloud application.
T_2	React UI	react, screen, navigation, issue, native	User interface issues, mainly with React navigation.
T_3	Styling	page, file, issue, version, app	General user interface styling.
T_4	VR and HAS	domoticz, revive, error, occulus, game	Errors mainly with LibreVR and Domoticz.
T_5	Methods	function, type, string, return, float	Method level compilation, type, and argument errors.
T_6	Features	time, issue, code, add, support	Suggestions or feature requests.
T_7	PHP and DB	php, wallabag, table, database, symfony	Hypertext preprocessor and database issues, mainly within the Wallabag application.
T_8	Presto	error, presto, java, jar, null	Java issues, mainly within the Presto application.
T_9	Logs	module, cljs, shadow, icinga, modules	Issues including log messages.
T_{10}	Nextcloud Sharing	www, lib, var, nextcloud, oc	Errors from Nextcloud's sharing functionality.
T_{11}	Node	npm, node, node_modules, error, js	Issues from Node.js and its package manager npm.
T_{12}	Build and Installation	version, file, install, docker, run	Issues with building and installing applications, generally from the command line.
T_{13}	Html and Latex	class, latexml, html, div, text	Display issues with html and latex elements.
T_{14}	Connectivity	info, debug, error, src, connection	Issues connecting to a service.
T_{15}	Notifications/BigchainDB	email, notifications, reply, bigchaindb, view	Issues using GitHub's notification service, and BigdbChainDB issues.
T_{16}	Ruby	lib, gems, logstash, ruby, usr	Issues with Ruby and ruby gems.
T_{17}	Plugins	plugin, node, jstree, function, android	Issues with plugins, mainly with JSTree plugin.
T_{18}	Build	build, src, include, usr, error	Issues with building applications.
T_{19}	Python	python, line, file, lib, packages	Issues with applications using Python.
T_{20}	HTTP	server, user, client, error, http	Connectivity, requests, and routing issues.

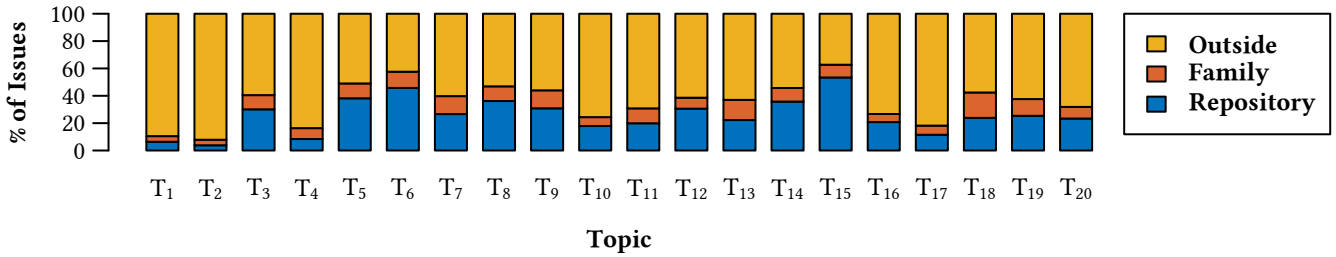


Fig. 5: Breakdown of each topic based on user affiliation.

bots or applications which set a common format or precedent for PRs, with P_6 and P_9 mostly coming from within the repository. For example, most PRs with topic P_8 `Coveralls` use the Coveralls application, which replies to the PR with the test code coverage that would result from accepting the PR. P_9 is not application specific; we see from Figure 3 that most mentions occur within repositories, which may explain why P_9 also occurs mostly within the repository.

P_2 `Additions`, P_{10} `Server Time & Logs`, and P_{13} `Builds` are all topics discussed across many repositories in our sample, with 248, 220, and 235 families out of 384 families discussing these topics, respectively. Interestingly, they all follow similar distributions to the number of PR topics within the repository and with the family, suggesting that there is little difference between discussions from PRs within a repository and PRs with the family.

Users report issues discussing different topics depending on whether they are from the repository, family, or outside the family. Users discuss similar topics in PRs from within the repository and from the family.

IV. Related Work

There are several studies showing the relationship between GitHub communication metrics and star count. Bissyand *et al.* show that there is a correlation between the number of reported issues and the number of watchers and forks in a repository [6]. Gousios *et al.* show that there is some correlation between the number of PRs a repository receives and its number of stars [39]. Borges *et al.* show there is a low correlation between stars and the number of commits, and moderate correlation between stars and the number of contributors and forks [5]. Blincoe *et al.* show that when popular users star, fork, contribute to or create a new repository, many of their followers will also star that repository [7]. Guzman *et al.* find a weak positive correlation between the average of positive commit comments belonging to a repository and its stars [40]. We performed similar analyses by finding relationships between communication metrics and stars.

There is also research on the importance of social behaviors in GitHub. Dabbish *et al.* discuss the impact of social inferences on work coordination and learning technical skills [41]. Zhang *et al.* show the benefits of @mentions to PR processing by engaging more users [27]. Tsay *et al.* identify several

TABLE VI: Descriptions for each PR topic.

Topic	Topic Name	Top Word Probabilities	Brief Description
P_1	BloomDesktop	src, bloomdesktop, bloombooks, file, reviews	PRs for the BloomDesktop application.
P_2	Additions	add, type, method, function, code	Adding new functionality.
P_3	Reviewable	files, reviewable, reviewed, img, alt	PRs using Reviewable, a code review application.
P_4	YCMD	ycmd, vallowic, reviews, file, line, completers	PRs surrounding YCMD, a code-completion server.
P_5	Google Bot and PRs	pull, branch, request, git, version	Google bot, or PRs talking about GitHub's functionally such as PRs, forking, merging, and branching.
P_6	Codecov	src, diff, pr, el, tree	PRs using Codecov.io, a code-coverage application.
P_7	React CI and Benchmarking	php, channel, mb, react, ms	React-navigation-ci and speed benchmarking.
P_8	Coveralls	coverage, status, badge, pulling, code	PRs using Coveralls, a testing code coverage application.
P_9	Mentionbot/Snyk-bot	user, files, pr, request, app	PRs using Mentionbot and Snyk-bot.
P_{10}	Server Time & Logs	code, time, patch, test, server	PRs surrounding time (delays, timeouts, rates etc.) and logging, mainly with servers.
P_{11}	Pentaho	pentaho, java, src, org, osgi	PRs with Pentaho's Open Source OSGI bundles.
P_{12}	Tables	fix, test, update, tests, add	PRs that use a checklist, table, or common format.
P_{13}	Builds	file, build, pr, docker, windows	PRs mentioning builds.
P_{14}	Python	python, pr, error, it, gt	PRs discussing updating Python dependencies.

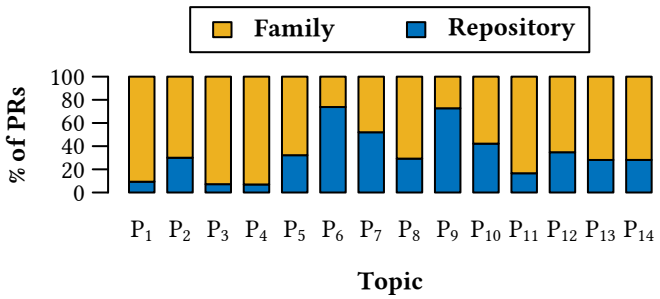


Fig. 6: Breakdown of each topic based on PR type.

factors that influence PR acceptance such as the number of comments, and social connections between the contributor and project manager [42]. Qiu *et al.* study the role of social capital on continued repository contribution [43].

Lastly, there is also research studying mainline variants and its forked variants in the context of Android apps on the Google Play store [44]. While this paper proposes an *app family*, a collection of applications on the Google Play store comprised of associated parents and children on GitHub, a software family, as defined in this paper, broadens the definition to fit the context of software development.

Our study combines the analyses of GitHub stars with the importance of studying communication on GitHub by separating communication metrics in related sets of repositories. The fact that we are performing these analyses by breaking down communication metrics through the lens of a software family distinguishes our work from previous studies.

V. Threats to Validity

We mined comments from the GitHub API and matched them to their corresponding comment ID in GHTorrent, as described in Section II. However, these comments could have been edited between the time of the database snapshot (March 1st, 2019) and the time of mining. This may have affected our sentiment analysis, as the comment could have been edited.

SentiStrength-SE is limited to English comments only. Comments in a different language that may have had a positive or negative score were excluded from our analysis, which may have further changed the results from our sentiment scores. Furthermore, despite being the most accurate sentiment analysis tool for software engineering texts, SentiStrength-SE is not 100% accurate, and may have brought a degree of error into our calculations [16].

We manually named and described each topic by looking at the top 5 word probabilities and a random sample of 10 issues and PRs. These names and descriptions were then verified against an independent source until agreement was reached. While this process was useful for defining our topics, it is ultimately subjective, and open for interpretation.

Lastly, we treated each observation (repository in our sample) as independent, which is a required assumption for linear regression models. However, this study has shown that families have a large influence on repository star count. Future studies should keep this in mind when analyzing GitHub repositories.

A replication package can be found in our GitHub repository [45].

VI. Conclusion

In this paper, we defined the concept of a software family to explore how interactions take place within and between repositories. We found that interactions from users in the same software family share a significant relationship with repository stars. We also found that issue discussions vary depending on who reported the issue relative to the family. Our results demonstrate that a software family is an important concept for investigating user contributions and repository stars.

For our future work, we plan to trace the evolution of software families with stars over time. We also plan on looking at developer retention and social capital among developers as they relate to families [43]. Lastly, we will investigate how different communication topics may relate to repository stars.

Acknowledgements

This research was funded in part by an NSERC Strategic Partnership Grant. We also thank Yuyang Liu for his help throughout this work, and anonymous reviewers whose feedback improved the presentation of our results.

REFERENCES

[1] GitHub, "The state of the octoverse," 2018. [Online]. Available: https://octoverse.github.com/

[2] GitHub, "Fork a repo," 2019. [Online]. Available: https://help.github.com/en/articles/fork-a-repo

[3] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[4] M. Biazzini and B. Baudry, "May the fork be with you: novel metrics to analyze collaboration on github," in *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics*. ACM, 2014, pp. 37–43.

[5] H. Borges and M. T. Valente, "What's in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

[6] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillere, J. Klein, and Y. Le Traon, "Got issues? who cares about it? a large scale investigation of issue trackers from github," in *2013 IEEE 24th international symposium on software reliability engineering (ISSRE)*. IEEE, 2013, pp. 188–197.

[7] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on github," *Information and Software Technology*, vol. 70, pp. 30–39, 2016.

[8] GitHub, "Saving repositories with stars," 2019. [Online]. Available: https://help.github.com/en/github/getting-started-with-github/saving-repositories-with-stars

[9] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2016, pp. 334–344.

[10] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley & Sons, 2013, vol. 751.

[11] O. J. Dunn, "Multiple comparisons using rank sums," *Technometrics*, vol. 6, no. 3, pp. 241–252, 1964.

[12] G. Gousios and D. Spinellis, "Ghtorrent: Github's data from a firehose," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 12–21.

[13] GitHub, "What happens to forks when a repository is deleted or changes visibility?" 2019. [Online]. Available: https://help.github.com/en/articles/what-happens-to-forks-when-a-repository-is-deleted-or-changes-visibility

[14] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of clinical epidemiology*, vol. 49, no. 12, pp. 1373–1379, 1996.

[15] M. Lui and T. Baldwin, "langid. py: An off-the-shelf language identification tool," in *Proceedings of the ACL 2012 system demonstrations*. Association for Computational Linguistics, 2012, pp. 25–30.

[16] M. R. Islam and M. F. Zibran, "Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text," *Journal of Systems and Software*, vol. 145, pp. 125–146, 2018.

[17] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 1, pp. 163–173, 2012.

[18] R. Berrios, P. Totterdell, and S. Kellett, "Eliciting mixed emotions: a meta-analysis comparing models, types, and measures," *Frontiers in Psychology*, vol. 6, p. 428, 2015.

[19] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding watchers on github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 336–339.

[20] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, and L. Zhang, "Why and how developers fork what from whom in github," *Empirical Software Engineering*, vol. 22, no. 1, pp. 547–578, 2017.

[21] H. Borges, M. T. Valente, A. Hora, and J. Coelho, "On the popularity of github applications: A preliminary note," *arXiv preprint arXiv:1507.00604*, 2015.

[22] O. Jarczyk, B. Gruszka, S. Jaroszewicz, L. Bukowski, and A. Wierzbicki, "Github projects. quality analysis of open-source software," 11 2014, pp. 80–94.

[23] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: Github as a collaborative social network," *ArXiv*, vol. 1407.2535, 2014.

[24] J. Fox and S. Weisberg, *An R Companion to Applied Regression*, 3rd ed. Sage, 2019.

[25] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2019. [Online]. Available: https://www.R-project.org/

[26] D. H. Ogle, P. Wheeler, and A. Dinno, *FSA: Fisheries Stock Analysis*, 2019. [Online]. Available: https://github.com/droglenc/FSA

[27] Y. Zhang, G. Yin, Y. Yu, and H. Wang, "A exploratory study of @-mention in github's pull-requests," in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1, Dec 2014, pp. 343–350.

[28] F. E. Harrell Jr, with contributions from Charles Dupont *et al.*, *Hmisc: Harrell Miscellaneous*, 2019. [Online]. Available: https://CRAN.R-project.org/package=Hmisc

[29] E. Noei, M. D. Syer, Y. Zou, A. E. Hassan, and I. Keivanloo, "A study of the relation of mobile device attributes with the user-perceived quality of android apps," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3088–3116, Dec 2017.

[30] S. J. Rao, "Regression modeling strategies: With applications to linear models, logistic regression, and survival analysis," *Journal of the American Statistical Association*, vol. 98, no. 461, pp. 257–258, 2003.

[31] M. J. Lee, B. Ferwerda, J. Choi, J. Hahn, J. Y. Moon, and J. Kim, "Github developers use rockstars to overcome overflow of news," in *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '13. New York, NY, USA: ACM, 2013, pp. 133–138.

[32] H. Borges, A. Hora, and M. T. Valente, "Predicting the popularity of github repositories," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM, 2016, p. 9.

[33] U. Grömping, "Relative importance for linear regression in r: The package relaimpo," *Journal of Statistical Software*, vol. 17, no. 1, pp. 1–27, 2006.

[34] M. E. Roberts, B. M. Stewart, D. Tingley *et al.*, "stm: R package for structural topic models," *Journal of Statistical Software*, vol. 10, no. 2, pp. 1–40, 2014.

[35] J. Silge and D. Robinson, "tidytext: Text mining and analysis using tidy data principles in r," *JOSS*, vol. 1, no. 3, 2016.

[36] R. Deveaud, E. SanJuan, and P. Bellot, "Accurate and effective latent concept modeling for ad hoc information retrieval," *Document numérique*, vol. 17, no. 1, pp. 61–84, 2014.

[37] M. Nikita, "Select number of topics for lda model," 2019. [Online]. Available: https://cran.r-project.org/web/packages/ldatuning/vignettes/topics.html

[38] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu, "A practical algorithm for topic modeling with provable guarantees," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 2. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 280–288.

[39] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 345–355.

[40] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: An empirical study," *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 05 2014.

[41] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 1277–1286.

[42] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 356–366.

[43] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going farther together: The impact of social capital on sustained participation in open source," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, May 2019, pp. 688–699.

[44] J. Businge, M. Openja, S. Nadi, E. Bainomugisha, and T. Berger, "Clone-based variability management in the android ecosystem," 09 2018, pp. 625–634.

[45] S. Brisson, E. Noei, and K. Lyons, "Replication package," 2019. [Online]. Available: https://github.com/idb-lab/family-replication-package