On the Shoulders of Giants: A New Dataset for Pull-based Development Research

Xunhui Zhang
National University of Defense Technology, Changsha, China.
zhangxunhui@nudt.edu.cn

Ayushi Rastogi
Delft University of Technology, the Netherlands
a.rastogi@tudelft.nl

Yue Yu
National University of Defense Technology, Changsha, China.
yuyue@nudt.edu.cn

ABSTRACT

Pull-based development is a widely adopted paradigm for collaboration in distributed software development, attracting eyeballs from both academic and industry. To better study pull-based development model, this paper presents a new dataset containing 96 features collected from 11,230 projects and 3,347,937 pull requests. We describe the creation process and explain the features in details. To the best of our knowledge, our dataset is the most comprehensive and largest one toward a complete picture for pull-based development research.

CCS CONCEPTS

• **Software and its engineering** → *Programming teams.*

KEYWORDS

pull-based development, pull request, distributed software development

ACM Reference Format:
Xunhui Zhang, Ayushi Rastogi, and Yue Yu. 2020. On the Shoulders of Giants: A New Dataset for Pull-based Development Research. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387489

1 INTRODUCTION

The pull-based development model [7] has changed the traditional way of code contribution [31], code review [28] and process automation [27]. Since Gousios et al. [8] proposed a firsthand dataset of pull request, plenty of valuable studies have been designed based on it, to better understand the essence of modern software development, *e.g,* human aspect of SE, DevOps and collaborative environment. Meanwhile, those studies demonstrate considerable extended features, *e.g,* gender [25], social connection [5], geographical location [18, 19], personality [10] and emotion[11], etc. However, there lacks a comprehensive dataset towards a more complete picture to support new work investigation and prior work reproduction and verification for pull-based development.

MSR '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387489

In this paper, standing on the shoulders of giants [6, 8], we create a new upgraded dataset, called **new_pullreq** (10 times larger than the original one) by adding all new features, as many as possible to the pool of existing metrics. To the best of our knowledge, our **new_pullreq** is the largest dataset for pull-based development research, which contains 11,230 OSS projects (representative of both small and large projects), 96 metrics and 3,347,937 pull requests. Our dataset is publicly available[1] and source code[2] is open source for replication as well as extension.

2 FEATURE SELECTION

The feature selection is based on Gousios et. al's dataset [8] as well as studies on pull request development from 2009 until 2019. By combining "pull-based development", "pull-request", "Github", "open source" with zero or more of the following sub-terms: "model", "software", "accepted", "rejected", "review", "merged", we searched the paper title using Google Scholar's boolean search engine, and identified 76 papers, a subset of which presented features for decision making. These features broadly falls into three categories: relating to contributor, project as well as pull request but some features lie at their intersection. Below we describe all 69 new features in addition to the 27 features reported in Gousios et. al's dataset [8].

2.1 Contributor characteristics

Contributor characteristics relate to submitters (or developer) and integrator (or committer). Some of the factors relate to individuals while others are interactions between two contributors or a contributor and a project.

- *Experience* of developers, conceptualized as the count of previous pull requests, previous pull request acceptance rate [8], accepted commit count [12], as well as days since account creation [17], can influence the pull request acceptance. First pull requests are less likely to be accepted [20, 21]. Similarly, experience of integrators calculated as the count of prior reviews influence decision making [2].
- *Core member*'s pull request are more likely to be accepted [1, 3, 16, 21, 24, 29].
- *Response time* of an integrator, often measured as the time to first response, likely influences the latency as well as chances of pull request acceptance [29].
- *Gender* of developers, when identified as female, reduces the chances of pull request acceptance [23].
- *Country* of developers influences pull request acceptance rate differently for different countries [19]. Further, if developer

[1] https://zenodo.org/record/3700595#.XmS2GJP0kY1
[2] https://github.com/zhangxunhui/new_pullreq_msr2020

and integrator are from the same country the chances of pull request acceptance increases [19].
- *Affiliation* of developers and integrators to companies as well as belongingness of both the developer and committer to the same company changes the chances of pull request acceptance [2, 14].
- *Personality* of developers and integrators individually as well as difference in personalities between the two, influence decision making [10]. Here, personality is conceptualized as Openness to Experience, Conscientiousness, Extraversion, Agreeableness, and Neuroticism (or OCEAN) and differences in personality as the difference between respective scores. For example, Extraversion_submitter - Extraversion_integrator.
- *Emotion* of developers as well as integrators, characterized as the percentage of positive and negative emotions, as well as the emotion of first comment are found to influence acceptance decision [11].
- *Social distance* refers to the closeness of code submitter to the potential integrator as well as the project. Following the integrator as well as the project prior to code contribution is seen to positively influence pull request decision making [24]. Relatedly, the fraction of team members who interacted with a developer over the team size in the last three months is used as a signal of social strength/trust, which increases the chances of pull request acceptance [29].

2.2 Project characteristics

- *Programming languages* tend to have different pull request acceptance rate [15, 17, 21]. For example, pull requests in Java and Python have less chance of acceptance and the opposite for Scala and R.
- *Popularity* of project, measured as watcher count [8], star count [8], and fork count [13, 17], negatively influences pull request acceptance [13, 17, 24].
- *Age* of project measured as the time interval between project creation and pull request creation (measured in months), indicates maturity of project as well as the less likelihood of pull request acceptance [24, 29].
- *Workload* of a project, as inferred from the number of open pull requests decreases the chance of pull request acceptance [2, 29].
- *Activeness* of project, as inferred from the time interval in seconds between the opening time of two latest pull requests, influences pull request acceptance [13].
- *Openness* of a project as inferred from the count of open issues as well as the pull request acceptance rate increases the likelihood of pull request acceptance [13].

2.3 Pull request characteristics

- *Size of change* is measured at commit-level (number of commits), file-level (files added, deleted, modified and changed) as well as type of files changed (source, document and others). Some of these metrics are coarse-grained, only discussing change (like source and test churn) while other metrics separate churn into addition and deletion [29]. Typically, increase in size reduces the chances of acceptance and vice-versa.
- *Complexity* of a pull request as inferred from the length of description is seen to negatively influence pull request acceptance [29].
- *Nature of pull request* as bug fix, for example, can increase the chances of PR acceptance [12, 15].
- *Test inclusion* of pull requests increase the chances of its acceptance[16, 24, 29].
- *Reference* of a contributor, issue or pull request can increase the change of pull request acceptance [4, 29].
- *Conflict* of a pull request, as explicitly mentioned in comments [7] negatively influences the chances of pull request acceptance.
- *Hotness* or relevance of a PR as inferred from the number of comments during code review process is seen to influence decision making [7, 12, 14, 20, 24, 29]. In addition to the issue comment count [8] and commit comment count [8], we add pull request comment count. Another indicator of hotness - number of participants [8] is also updated to reflect participation in issues, commits, and pull requests.
- *Emotions* (positive, negative and neutral) surrounding a pull request discussion reflect reviewer's reaction and is found to influence decision making [11].
- *Continuous Integration* of a pull request (its existence or not), latency, build count, all tests passed, percentage of tests passed/failed, first and last build status are all seen to influence pull request decision making [9, 22, 27, 29, 30].

The summary of each category of **new_pullreq** is shown in Table 1, which includes feature tag, description and related citations. The mysql table structure [3] and technical report [4] can be seen in the Github project.

3 DATASET

Similar to the previous study by Gousios et al. [8], the new dataset for pull-based development research builds on the publicly available datasets hosted on GHTorrent[5]. We use the latest version of Mysql data dump [6] (created on 1 June 2019) and complement it with additional information (e.g. issue comments) provided in the comparable version of MongoDB dump [7].

To create a large dataset of active and representative software repositories, we applied several inclusion and exclusion criteria.

(1) We selected all source (base) repositories and removed forks or otherwise deleted repositories from GHTorrent dataset or GitHub. Forks with shared history as the source repository can influence the representativeness while deleted repositories are not active anymore. Further, to select actively developed repositories, we included repositories with new pull requests in the last three months.

[3] https://github.com/zhangxunhui/new_pullreq_msr2020/blob/master/table_structure.pdf
[4] https://github.com/zhangxunhui/new_pullreq_msr2020/blob/master/technical_report.pdf
[5] http://ghtorrent.org/
[6] http://ghtorrent-downloads.ewi.tudelft.nl/mysql/mysql-2019-06-01.tar.gz
[7] http://ghtorrent-downloads.ewi.tudelft.nl/mongo-daily/mongo-dump-2019-06-30.tar.gz

Table 1: Factors influencing pull-based development

Feature	Description	Feature	Description
Contributor Characteristics			
acc_commit_num	The number of accepted commits of a contributor before the creation of a pull request[12]	account_creation_days	The time interval in days from the contributor's account creation to the pull request creation[17]
first_pr	Whether it is the first pull request of a contributor[20, 21]	prior_review_num	The number of prior reviews of an integrator[2]
core_member	Whether the contributor is a core member or not[1, 3, 16, 21, 24, 29]	first_response_time	The time interval in minutes from pull request creation to the first response by a reviewer[29]
contrib_gender	The gender of a contributor[23]	contrib/inte_country	Country of residence of contributor/integrator[19]
same_country	Whether contributor and integrator come from the same country[19]	prior_interaction	Number of times that the contributor interacted with the project in the last three months[24]
same_affiliation	Whether the contributor and the integrator belong to the same affiliation[2, 14]	contrib/inte_affiliation	The affiliation that the contributor/integrator belongs to[2, 14]
contrib/inte_X	The Big Five personality traits of contributor/integrator (open: openness; cons: conscientious; extra: extraversion; agree: agreeableness; neur: neuroticism)[10]	perc_contrib/inte_X	The percentage of contributor/integrator's emotion in comments (neg: negative/pos: positive/neu: neutral)[11]
X_diff	The absolute difference of Big Five personality traits between contributor and integrator[11]	contrib/inte_first_emo	The emotion of the contributor/integrator's first comment[11]
social_strength	The fraction of team members that interacted with the contributor in the last three months[29]	contrib_follow_integrator	Whether the contributor follows the integrator when submitting a pull request[24]
Project Characteristics			
language	Programming language of project[15, 17, 21]	open_issue_num	Number of opened issues when submitting the pull request[13]
project_age	Time interval in months from the project creation to the pull request creation[24, 29]	open_pr_num	Number of opened pull requests when submitting the pull request[2, 29]
pushed_delta	The time interval in seconds between the opening time of the two latest pull requests[13]	fork_num	Number of forks of project when submitting the pull request[13, 17]
pr_succ_rate	Acceptance rate of pull requests in the project[13]		
Pull Request Characteristics			
churn_addition	Number of added lines of code[29]	churn_deletion	Number of deleted lines of code[29]
bug_fix	Whether pull request fixes a bug[12, 15]	description_length	Word count of pull request description[29]
test_inclusion	Whether test code exists in a pull request[16, 24, 29]	comment_conflict	Whether the keyword "conflict" exists in comments[7]
hash/at_tag	Whether #/@ tag exists in comments or description[4, 29]	pr_comment_num	Number of pull request comments[8]
part_num_X	Number of participants in comment (issue: issue comment; pr: pull request comment; commit: commit comment)[8]	part_num_code	Number of participants in both pull request comment and commit comment[8]
ci_exists	Whether a pull request uses continuous integration tools[27]	ci_build_num	Number of CI builds[30]
ci_latency	Time interval in minutes from pull request creation to the first build finish time of CI tools[29]	perc_neg/pos/neu_emotion	Percentage of negative/positive/neutral emotion in comments[11]
ci_test_passed	Whether passed all the CI builds[9, 22]	ci_first_build_status	First build result of CI tool[30]
ci_failed_perc	Percentage of failed CI builds[30]	ci_last_build_status	Last build result of CI tool[30]

(2) Next, we select projects from six programming languages, different in size and activity count for meaningful analysis. We selected all projects with at least 33 submitted pull requests. These projects constitute top 3% of all projects in terms of pull request count (as against top 1% in case of Gousios et. al's [8] dataset). We extended original selection of four programming languages (Ruby, Python, Java, and Scala) by Go and Javascript. The resulting 19,572 projects were distributed across projects as follows: Javascript: 6,584; Python: 5,121; Java: 3,044; Ruby: 2,794; Go: 1,497; and Scala: 532. Next, we selected different-sized projects (small, medium, and large) in terms of contributor count. The selected small teams comprised of 12 or less developers, medium-sized teams with 13 and up to 30 developers, and large teams with more than 30 developers. We selected 4,000 projects from each class, resulting in a total of 12,000 projects. Among these projects, we removed "everypolitician/everypolitician-data" which is extremely large, and is used for holding the data for national legislatures worldwide. Moreover, a large fraction of the activities on this project are through bots.

(3) Finally, at the pull-request level, we included all pull requests that were submitted to the default branch of the repository and are not open otherwise (no decision is being made on open pull requests). Moreover, we remove those projects that have less than 20 default branch related closed pull requests. This gives us 11,230 projects comprising of 3,347,937 pull requests. In comparison to Gousios et. al's dataset of 865 projects and 336,502 pull requests, our dataset has 12 times more projects but only about 10 times more pull requests (since we also included small projects).

4 FEATURE COLLECTION

For extracting features from data, we followed the procedure specified in the respective paper. We retained all variants of a feature proposed in literature, with a few exceptions (like emotion and personality) discussed in the below. There were, however, situations where we had to extrapolate the solution for representativeness. For example, the existing solution to analyze continuous integration works only for TravisTorrent. To make our dataset generalizable, we expanded the existing solution with some heuristics.

Personality Many models of personality are available in literature and used in existing research. For this dataset, we choose the state-of-the-practice tool - *IBM Watson Personality Insights* [8] - to measure the Big Five Personality Traits of each user [10]. We collected all comments of developers from issue discussions, pull request discussions as well as commit discussions. We processed the data to remove code snippets and special characters (including quotes, # tags, @, IPs, email address, URLs, and numbers) which otherwise are of no use to infer personality. The resulting data is feed as input to the Personality Insights tool conditioned only on the availability of 100 or more words as input to ensure a sizable text for reliable interpretation.

[8] https://www.ibm.com/watson/services/personality-insights/

Country and Gender We infer country and gender of developers using the tool proposed by Vasilescu et al. [25].

Emotion Similar to personality, many models exist to infer emotions. We use the best prediction model, known so far - UmlFit [11] to infer emotions in discussions.

Continuous integration The previous studies used only one tool, travis-ci [27]. However, for whether a pull request uses CI tool, using travis-ci only meant that the existing solution no longer holds. To overcome this challenge, we proposed a few heuristics applicable to a wider range of CI tools.

To find whether a pull request uses a CI tool or not, we started by searching for terms commonly used for continuous integration such as "continuous", "integration", "-ci", "ci-", "ci/", "ci." in the text fields of pull requests. These fields include context, description as well as the associated URL, information on which are inferred from GitHub status API [9]. If a term match is found, the pull request uses CI tool, otherwise we look for keywords "build" and "test" in the context and description. Alternatively, we compiled a list of widely used CI tools from GitHub marketplace [10] and verified it manually by looking at related posts online. Further, we checked for the presence of tool name in text fields as a sign of CI tool use. We assume that if the above steps did not link a pull request to a CI tool, CI tool is not used. To check for the accuracy of the proposed heuristic, the first author randomly selected 200 pull requests inferred using CI tools and another 200 pull requests not inferred using CI tools. The first author then manually checked all 400 pull requests and found 99.5% precision for pull requests that uses CI tools and 99% precision for pull requests that do not use CI tools.

For other metrics, such as CI build count and percentage of failed CI, in order to make the dataset more generalizable, we present insights from three widely used CI tools: travis-ci, circle-ci and drone-ci. For travis-ci, we use the method proposed by Vasilescu et al. [26] to retrieve CI related metrics. For the other two CI tools, unlike travis-ci, there did not exist a direct link between a pull request and CI tool. To link a pull request to a CI tool, we used repository slug "username/repo" as an argument and matched the commit sha of each build with a pull request. If such a match exists, we link a build with a pull request.

Affiliation Studies on affiliation as a feature examined well-known repositories, the solution, however, was not generalizable. In this study, we introduced a new approach to infer affiliation from company and email domain information derived from GitHub API. First, we select the company texts that appear more than 10 times in the dataset. By manually checking it, the first author identify a list of stop words including freelancer, student, and remove it. Next, we look for university affiliation by mapping both the university name and its abbreviation to university. All other affiliations are seen as related to a company. To filter company name, we removed prefix "@" and suffixes such as "ltd." and "corp.". Further, we changed some names to its alias. For example, aws to amazon and qihoo to 360.

We further enrich our inference of affiliation using email domain. We identified world popular domains[11], list of world university domains[12]. We removed world popular domains as they cannot identify company affiliation uniquely. Next, we mapped university email domain to university affiliation. For all other email domains, followed by ".org" or ".com", we mapped them to company. We also defined some stop words including gmail, github, yahoo and removed them, as some domains are missed in the popular domain list. If an email domain uniquely maps to an alias (with at least 30 data points to avoid false positives), we append the known affiliation to the affiliation inferred from company text.

[9]https://developer.github.com/v3/repos/statuses/
[10]https://github.com/marketplace?category=continuous-integration
[11]https://github.com/mailcheck/mailcheck/wiki/List-of-Popular-Domains
[12]https://github.com/Hipo/university-domains-list

5 LIMITATIONS

With an objective to create a large and representative dataset for future research on pull-based development, we collected 96 features from 11K+ projects and 3 million+ pull requests. In the process, however, we made many choices or inherited it from previous studies that can impact the dataset. Our work builds on a decade of research on pull-based development, extracting features relevant for decision making. This way we not only stand on the shoulders of giants, and hence benefiting from it but also inherit limitations of the features they present. The methodology adopted in this study is similar to and builds on the original study by Gousios et al [8]. Similar to their work, we combine data from multiple sources: GHTorrent Mysql data dump, MongoDB data dump, as well as git repository data downloaded from GitHub. Since each of the three data sources have data at different levels of abstraction, this can lead to some differences in outcome. We, however, went an extra mile to improve the representativeness of the dataset. We added two new programming languages and extrapolated known features (*e.g.* continuous integration) to a variety of small and large projects. That being said, we realize that there can be a more representative dataset also including code-related metrics which otherwise are found not important for decision making and less explored or metrics that cannot be studied objectively.

6 RESEARCH OPPORTUNITIES

The pull-based model provides a synthesized paradigm for distributed collaboration, which has attracted global attentions in recent years. In this paper, we create a comprehensive and large-scale dataset collected from 11K+ representative OSS projects in GitHub, and describe the creation process and explain all features in details. Our dataset, in addition to supporting research on pull-based development, provides new opportunities to the related research fields, spanning from collaborative environments (*e.g.,* code patch and code review), software maintenance (*e.g.,* bug prediction), software process (*e.g.,* continuous integration and DevOps), human factors in computing systems (*e.g.,* developer personality) and etc. Researchers can achieve a more complete picture of distributed development by empirical study, and even train artificial intelligence models based on our carefully filtered data samples.

ACKNOWLEDGMENTS

This work is supported by Science and Technology Innovation 2030 of China (Grand No.2018AAA0102304), National Nature Science Foundation of China (Grand No.61702534) and China Scholarship Council. Thank you Dr. Georgios Gousios, Rahul N. Iyer, Frenk van Mil, Celal Karakoc, Leroy Velzel, Daan Groenewegen and Sarah de Wolf for your technical help.

REFERENCES

[1] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey. 2012. The Secret Life of Patches: A Firefox Case Study. In *2012 19th Working Conference on Reverse Engineering*. 447–455. https://doi.org/10.1109/WCRE.2012.54

[2] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey. 2013. The influence of non-technical factors on code review. In *2013 20th Working Conference on Reverse Engineering (WCRE)*. 122–131. https://doi.org/10.1109/WCRE.2013.6671287

[3] Amiangshu Bosu and Jeffrey C. Carver. 2014. Impact of Developer Reputation on Code Review Outcomes in OSS Projects: An Empirical Investigation. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Torino, Italy) *(ESEM âĂŹ14)*. Association for Computing Machinery, New York, NY, USA, Article Article 33, 10 pages. https://doi.org/10.1145/2652524.2652544

[4] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2017. A preliminary analysis on the effects of propensity to trust in distributed software development. In *2017 IEEE 12th international conference on global software engineering (ICGSE)*. IEEE, 56–60.

[5] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. 2015. Developer onboarding in GitHub: the role of prior social links and language experience. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 817–828.

[6] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories* (San Francisco, CA, USA) *(MSR '13)*. IEEE Press, Piscataway, NJ, USA, 233–236. http://dl.acm.org/citation.cfm?id=2487085.2487132

[7] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 345âĂŞ355. https://doi.org/10.1145/2568225.2568260

[8] Georgios Gousios and Andy Zaidman. 2014. A Dataset for Pull-Based Development Research. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 368âĂŞ371. https://doi.org/10.1145/2597073.2597122

[9] G. Gousios, A. Zaidman, M. Storey, and A. v. Deursen. 2015. Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 358–368. https://doi.org/10.1109/ICSE.2015.55

[10] R. N. Iyer, S. A. Yun, M. Nagappan, and J. Hoey. 2019. Effects of Personality Traits on Pull Request Acceptance. *IEEE Transactions on Software Engineering* (2019), 1–1. https://doi.org/10.1109/TSE.2019.2960357

[11] Iyer, Rahul. 2019. Effects of Personality Traits and Emotional Factors in Pull Request Acceptance. http://hdl.handle.net/10012/14952

[12] Y. Jiang, B. Adams, and D. M. German. 2013. Will my patch make it? And how fast? Case study on the Linux kernel. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. 101–110. https://doi.org/10.1109/MSR.2013.6624016

[13] Nikhil Khadke, Ming Han Teh, and Minghan Shen. [n.d.]. Predicting Acceptance of GitHub Pull Requests. ([n. d.]).

[14] O. Kononenko, T. Rose, O. Baysal, M. Godfrey, D. Theisen, and B. de Water. 2018. Studying Pull Request Merges: A Case Study of Shopify's Active Merchant. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. 124–133.

[15] Rohan Padhye, Senthil Mani, and Vibha Singhal Sinha. 2014. A Study of External Community Contribution to Open-Source Projects on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 332âĂŞ335. https://doi.org/10.1145/2597073.2597113

[16] Gustavo Pinto, Luiz Felipe Dias, and Igor Steinmacher. 2018. Who Gets a Patch Accepted First? Comparing the Contributions of Employees and Volunteers. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering* (Gothenburg, Sweden) *(CHASE âĂŹ18)*. Association for Computing Machinery, New York, NY, USA, 110âĂŞ113. https://doi.org/10.1145/3195836.3195858

[17] Mohammad Masudur Rahman and Chanchal K. Roy. 2014. An Insight into the Pull Requests of GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 364âĂŞ367. https://doi.org/10.1145/2597073.2597121

[18] Ayushi Rastogi. 2016. Do Biases Related to Geographical Location Influence Work-Related Decisions in GitHub?. In *Proceedings of the 38th International Conference on Software Engineering Companion* (Austin, Texas) *(ICSE âĂŹ16)*. Association for Computing Machinery, New York, NY, USA, 665âĂŞ667. https://doi.org/10.1145/2889160.2891035

[19] Ayushi Rastogi, Nachiappan Nagappan, Georgios Gousios, and André van der Hoek. 2018. Relationship between Geographical Location and Evaluation of Developer Contributions in Github. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Oulu, Finland) *(ESEM âĂŹ18)*. Association for Computing Machinery, New York, NY, USA, Article Article 22, 8 pages. https://doi.org/10.1145/3239235.3240504

[20] D. M. Soares, M. L. d. L. JÃžnior, L. Murta, and A. Plastino. 2015. Rejection Factors of Pull Requests Filed by Core Team Developers in Software Projects with High Acceptance Rates. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 960–965. https://doi.org/10.1109/ICMLA.2015.41

[21] Daricélio Moreira Soares, Manoel Limeira de Lima Júnior, Leonardo Murta, and Alexandre Plastino. 2015. Acceptance Factors of Pull Requests in Open-Source Projects. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (Salamanca, Spain) *(SAC âĂŹ15)*. Association for Computing Machinery, New York, NY, USA, 1541âĂŞ1546. https://doi.org/10.1145/2695664.2695856

[22] Y. Tao, D. Han, and S. Kim. 2014. Writing Acceptable Patches: An Empirical Study of Open Source Project Patches. In *2014 IEEE International Conference on Software Maintenance and Evolution*. 271–280. https://doi.org/10.1109/ICSME.2014.49

[23] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. 2017. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science* 3 (2017), e111.

[24] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of Social and Technical Factors for Evaluating Contribution in GitHub. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 356âĂŞ366. https://doi.org/10.1145/2568225.2568315

[25] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and Tenure Diversity in GitHub Teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI âĂŹ15)*. Association for Computing Machinery, New York, NY, USA, 3789âĂŞ3798. https://doi.org/10.1145/2702123.2702549

[26] B. Vasilescu, S. van Schuylenburg, J. Wulms, A. Serebrenik, and M. G. J. van den Brand. 2014. Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub. In *2014 IEEE International Conference on Software Maintenance and Evolution*. 401–405. https://doi.org/10.1109/ICSME.2014.62

[27] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA, 805âĂŞ816. https://doi.org/10.1145/2786805.2786850

[28] Y. Yu, H. Wang, G. Yin, and C. X. Ling. 2014. Who Should Review this Pull-Request: Reviewer Recommendation to Expedite Crowd Collaboration. In *2014 21st Asia-Pacific Software Engineering Conference*, Vol. 1. 335–342. https://doi.org/10.1109/APSEC.2014.57

[29] Yue Yu, Gang Yin, Tao Wang, Cheng Yang, and Huaimin Wang. 2016. Determinants of pull-based development in the context of continuous integration. *Science China Information Sciences* 59, 8 (2016), 080104. https://doi.org/10.1007/s11432-016-5595-8

[30] F. Zampetti, G. Bavota, G. Canfora, and M. D. Penta. 2019. A Study on the Interplay between Pull Request Review and Continuous Integration Builds. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 38–48. https://doi.org/10.1109/SANER.2019.8667996

[31] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. 2016. Effectiveness of code contribution: From patch-based to pull-request-based tools. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 871–882.