10X Engineers—Myth or Reality?

Min Hee Son, Hannah Brown, Nikita Andrikanis,
Spencer Pham, Andrew Theodorides
{mhson, hsbrown, nandrikanis, spham, atheodorides}@ucdavis.edu
ECS 189L, Group: Red Sus
UC Davis

December 2020

1 Introduction

10x engineers are said to be software engineers that are ten times better than the average programmers because they are ten times more talented and ten times more productive. The existence and characteristics of 10x engineers are topics of high interest, especially within developer communities. Even large corporations are interested in ways to objectively measure software development processes and individual programmer productivity [1, 2]. But are 10x engineers a myth or do they actually exist? And if they do, what differentiates them from everyone else?

We want to study this because although there is much conversation around the topic of productivity in software engineering [2, 3, 4], there is currently no universal, standardized measurement to assess and identify what makes a 10x engineer. Previous studies have looked at measures for entire software development units which do not give insight into individual programmers. Other studies have looked at self-reported measures of productivity which provide broad information that is easier to gather but is not necessarily confirmed by numerical, concrete data. Discovering the traits of a 10x engineer could help other programmers improve their work in specific, targeted areas and provide companies with valuable information in the pursuit of trustworthy and skilled engineers who can be trusted lead their development. Especially for start-ups with limited budgets and high stakes, finding a 10x engineer could significantly boost their chances of success.

2 Background

The question of software developer productivity has been investigated in a number of different methods and scope levels. In one study, researchers designed a survey to investigate what developers at three different companies believed contributed to their productivity [2]. Though this method was subjective, they sought to achieve breadth by allowing developers to self-report their productivity levels and what contributed to it. From this study, researchers found that job enthusiasm, peer support for new ideas, and useful feedback about job performance were the most statistically significant factors across all three companies. However, these factors were not all specific to developer-specific productivity but to other workers as well.

In another study conducted by researchers at Volkswagen, researchers were interested in finding key performance indicators which were defined as a "set of values allowing to analyze and compare an organization's performance regarding specified objectives" [1]. These key performance indicators (KPI's) would measure the existing processes in regards to software quality, efficiency, and stability. This task was quite difficult to carry out as the KPI's needed to meet a number of criteria that would make it applicable to various departments and easily understandable across organizational and expertise levels.

Outside of large organization or companies, some research has also been conducted to study individual programmers in hopes to more objectively measure variation in developer productivity. One study argues that the variation in productivity lies equally in performances differences between programmers and performance differences within one programmer's different assignments [5]. The results

from this study suggested that there was weak evidence for the existence of hyper-productive programmers and that there was a general uniformity in productivity across programmers. Especially in regards to completion time of an assignment, programmer productivity varied greatly between different assignments. This variation suggested that there may be a myriad of reasons behind the speed at which a task is completed, some which may be unrelated to true differences in programmer productivity.

3 Research Questions

We aim to discover the factors that 10X developers and the projects they work on share. We identified several possible factors to explore based on existing perceptions of what a highly productive engineer would look like. By answering these, we hope to answer the main question of whether or not the 10X engineer is a valid concept.

RQ 1: Do 10X engineers commit more or less frequently than others?

A common informal metric of how productive a programmer within a social programming settings such as GitHub is how often they commit. An all green contribution graph on GitHub is a common informal metric used to decide how good of an engineer someone is.

RQ 2: Do 10X engineers act as contributors or maintainers more often?

Another factor of interest when exploring productive engineers is whether or not the projects they contribute to have any significant differences in documentation. To explore whether the projects 10X engineers contribute to tend to be better documented than others, we ask:

RQ 3: Do the projects 10X engineers contribute to typically have more or less comments than those contributed to by non-10X engineers?

As a supplemental question, we also explore other statistics of the projects 10X engineers contribute to to see if there is any pattern to them.

RQ 4: What is the complexity and maintainability of the projects contributed to by 10X engineers, and what is the average length of their associated files?

4 Data

In the initial phase of our project, we collected two samples of python developers, the first (the random developer set) of one-thousand randomly sampled developers and the second (our "gold-star" developer set) of the top one-thousand developers ranked by how many stars they have on their python repositories. This list of developers ranked by stars was obtained by scraping the site Git Awards [6]. Though Git Awards includes companies and organizations in their list, we excluded these to obtain a list of only individual developers.

The login IDs for one-thousand random developers were collected from GHTorrent [7] using Google BigQuery. The database was queried for a set of ten-thousand random developers (more than one-thousand developer logins were collected to avoid overlap with the gold star set), and then a thousand random projects were randomly sampled from these developers' contributions.

After collecting both sets of IDs and projects, they were used to query the GHTorrent database for the commit information linked to each user. Commit information was collected for each user's contributions to a randomly chosen project. Projects with only one commit were filtered out for both the gold-star set and the random set.

Note however that we were soon able to query a data set containing every user with at least a star and their corresponding star count. Hence, there are some figures that will actually contain several percentiles instead of just the random users and top one-thousand users. For reference, the term "percentile" actually refers to a bin or a range. For example, the 5% percentile refers to the range 1% - 5% top users according to their star count.

5 Methods

After retrieving the data, a distribution was fit to the number of commits by each user from both the random and gold-star sets and the number of users. To explore the first question, for both the random group and the gold-star group, we plotted a histogram of user commits to a randomly selected project according to how many standard deviations from the mean they were (see figures Figs. 3 and 4). This allowed us to see if any patterns emerged between the two groups.

Because 10X engineers should appear as outliers, we also used two outlier detection algorithms on the commit data: Isolation Forest and Local Outlier Factor (LOF). We performed further analysis on the outliers identified by these algorithms to see if there was any hierarchy to the outliers—that is some developers who committed far more than others even in this exceptional group.

To explore the code in the projects that we sampled, we used Radon [8] to measure the Halstead complexity [9], number of comments, length in lines of code, and maintainability of the code in each project. These metrics were collected for 150 users randomly sampled from each of the percentile groups from the initial exploration of commits per user, with 30 files examined for each user (dependent on them having this many).

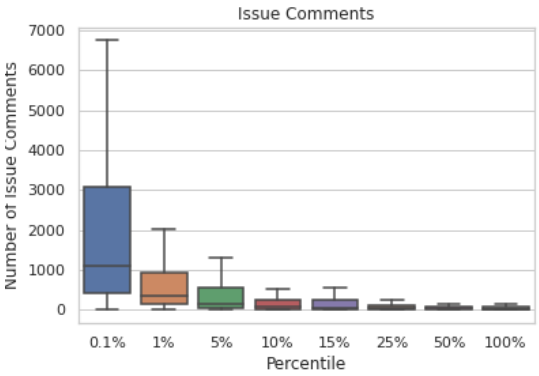
The code used to retrieve the data, produce the percentile plots of commits, identify outliers, and sample and analyze user projects with Radon is available at: https://github.com/hannah-aught/ecs189L-final-project.

6 Results

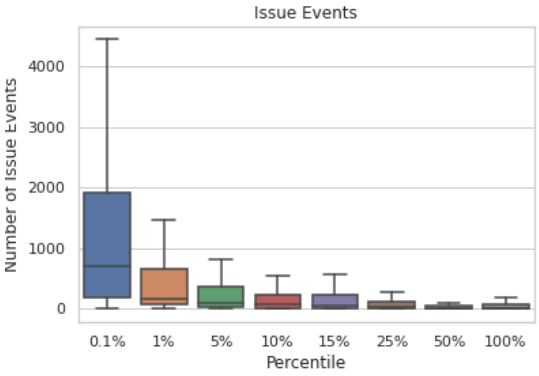
When exploring the distributions commits from developers in the two sets, we found that there was far less variance in the commits from the random developers than the number of commits for the gold-star developers—with or without outliers present (see Table 1). The variance was significantly higher in the original gold-star group with the outliers still present. This high variance led us to exploring outliers with LOF and isolation forest. As expected from initial EDA, there were more outliers in the gold-star set than the random set (see Table 2).

We were also able to analyze several metrics from GHTorrent that showed a distinct distribution: commit comments, commits, issue comments, issue events, issues reported, issues assigned, organization members, project members, projects, pull request comments, pull request history, and projects starred. For clarification about any table and what they mean, refer to the GHTorrent schema (ghtorrent.org/relational.html). As expected, all the metrics are heavily correlated with the star count of the user. However, we can still find metrics that had the largest correlations. At first, we hypothesized that commits were going to be the biggest differentiator for the 10x developers. Our assumption was that 10x developers likely outputted a lot more code (i.e. commiting) which would allow them to accumulate a lot of stars.

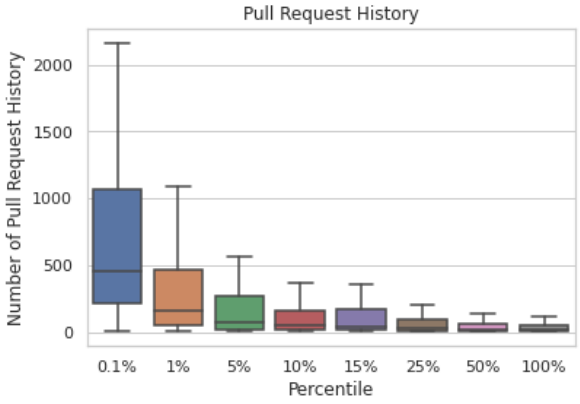
However the top four biggest factors in order were actually issue comments, issue events, pull request history, and commits. In all of the top four metrics, the .1% group had at least a 2000% increase of the median in that metric compared to the 100% group. For instance, the .1% group had 6500% more issue comments compared to the 100% group.



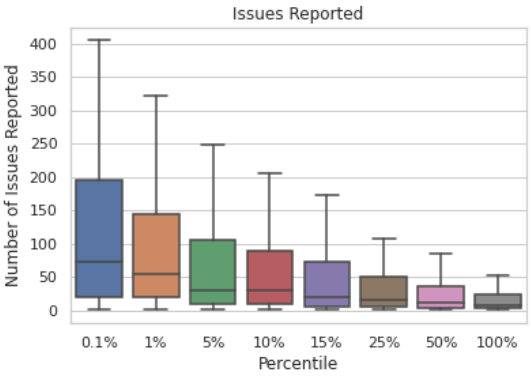
After analyzing the results, we now hypothesize that being able to recognize issues and quickly discuss them separates the top developers, not necessarily outputting as much code as possible.



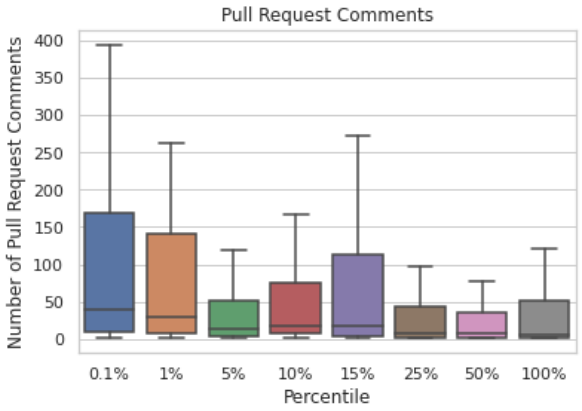
Again, being able to identify issues and fix them are the biggest signs of a 10x developer. Notably, the 0.1% group has nearly a 5000% increase in issue event activity. One hypothesis to support this is that repositories that are well-liked (i.e. have a high star count) will have a lot of issue event activity. Because of this, issues are actively being sought out and fixed in the repository, which causes users and clients to rate them highly.



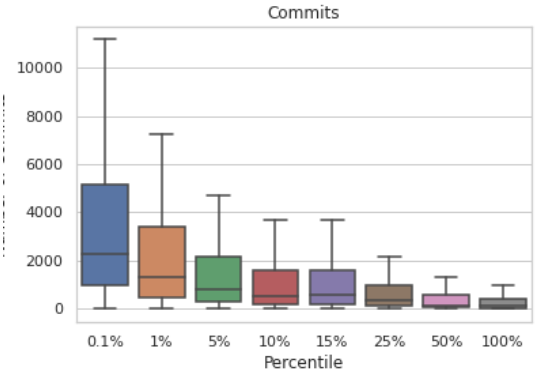
In order to have a lot of issues be fixed, a repository would likely have a lot of pull request activity. However, note that pull requests aren't just reserved for issues. Likely, these pull requests are implementing a lot of features that users will enjoy. As expected, the 0.1% group have a 3000% increase in pull request history compared to the baseline.



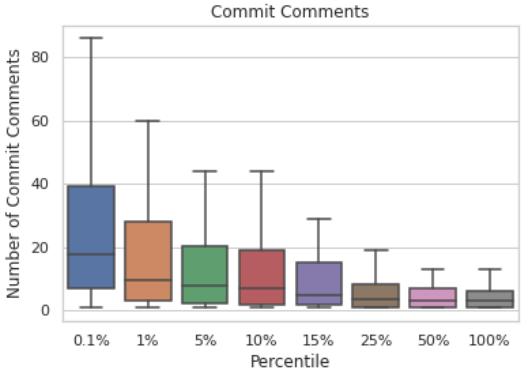
The top percentile group reported 850% more issues compared to the baseline. Issues still prove to be a big factor into making a repository well-liked. However, this metric still isn't as significant as issue comments or issue events, likely because reporting an issue doesn't actually remedy any known issues. Instead, it just adds to the list of known bugs.



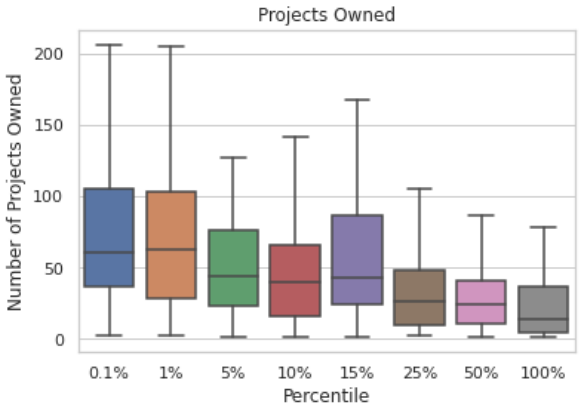
The top percentile group had 700% more pull request comments compared to the baseline.



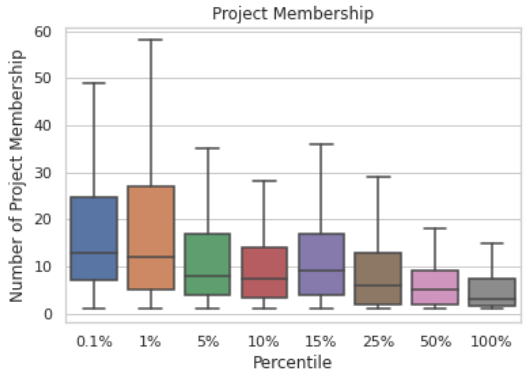
Although commits weren't as big of a factor as we anticipated, it still shows that they are a good indicator of a 10x developer. A 2000% increase from the baseline indicates that outputting a large quantity of code still helps.



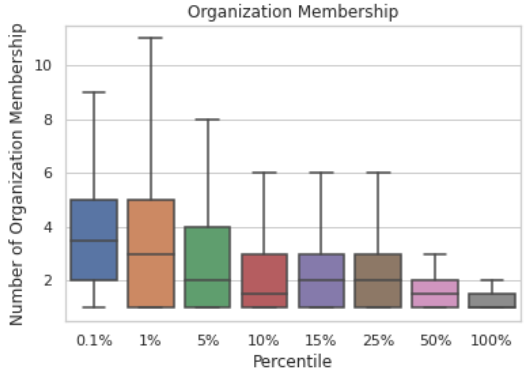
The top percentile group had 500% more commit comments compared to the baseline.



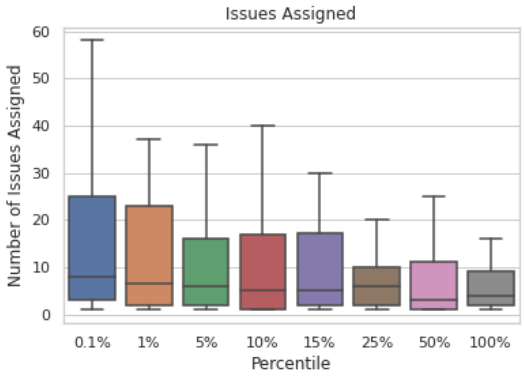
Perhaps the metric that was most surprising was projects owned. The top percentile only had 350% more projects compared to the baseline. Although one would expect that having a lot of projects would highly correlate with having a lot of stars, this does not happen to be the case. Instead, it seems that top developers have a few repositories that they put an extra amount of effort into, such as through issue events and issue comments.



The top percentile group were members of 325% more projects compared to the baseline.



The top percentile group were members of 250% more organizations compared to the baseline.



Finally, issues assigned is the least significant metric, with a 100% increase compared to the baseline. Although the reason is unknown, there are two possible guesses. One is that top developers like to work on issues by themselves without being advised. Another is that top developers are the only ones working on top starred repositories, so no one else is able to assign them issues.

	Variance
Random	12321
Gold-star	80766169
Gold-star (outliers removed)	22724289

Table 1: Variance for each of the commit groups

	Random	**Gold-Star**
LOF	128	199
Isolation Forest	31	15
Total Outliers	143	202

Table 2: Outliers found for each of the commit groups

Exploration of the outliers found in the gold-star set showed that not only were there more outliers (which can be seen comparing Figs. 3 and 4, but there was a hierarchy in the outliers as well, with some very extreme outliers (users with far more commits than the others) lying outside the main group of outliers. This can be seen in the box-plot in Fig. 1. A plot of the gold-star commit data grouped by standard deviations from the mean can be seen in Fig. 5. Though removing the outliers did bring the gold-star commit plot slightly closer to the random commit plot, we still see significant differences in the spread of the data.

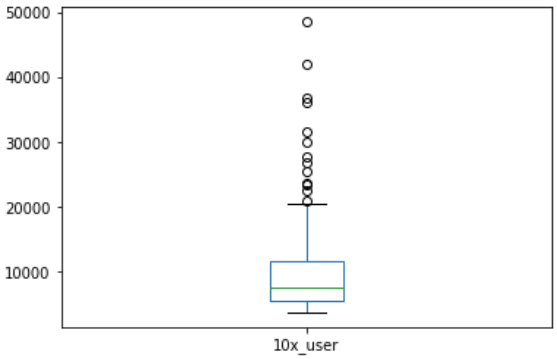


Figure 1: Box plot of only the outliers for number of commits for the gold-star developers

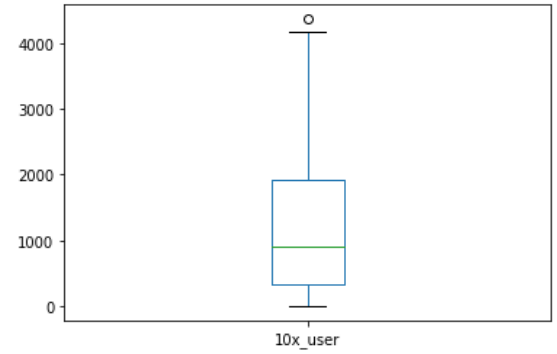


Figure 2: Box plot of the number of commits for the gold-star developers with outliers found using LOF and isolation forest excluded

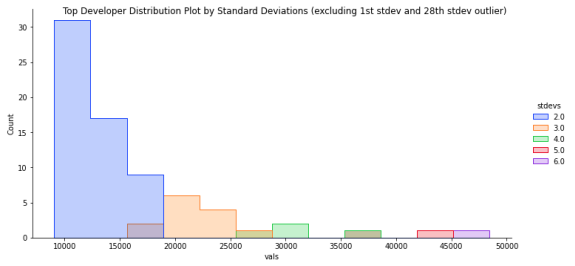


Figure 3: Commits for gold-star developers grouped by standard deviations above the mean with extreme outliers in the first and 28th standard deviation excluded

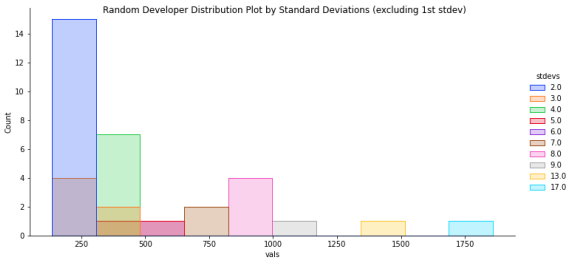


Figure 4: Commits for random developers grouped by standard deviations above the mean with extreme outliers in the first standard deviation from the mean excluded

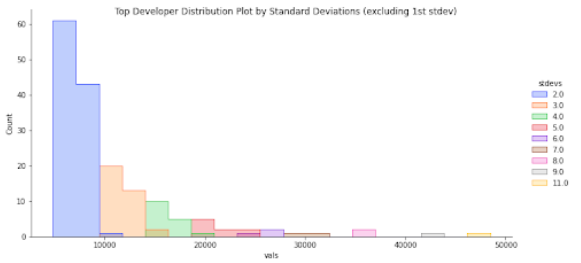


Figure 5: Commits for gold-star developers grouped by standard deviations above the mean with the first standard deviation excluded for scale

The results of using Radon to explore the complexity, maintainability, number of comments, and lines of code per file showed no major differences across the .1%, 1%, and 5% percentiles (Fig. 6). There do appear to be some differences in the number of comments per file across all percentiles, with the first percentiles having less comments per file on average (Fig. 7), but further exploration is needed to make a definitive statement about the coding tendencies of 10X engineers.

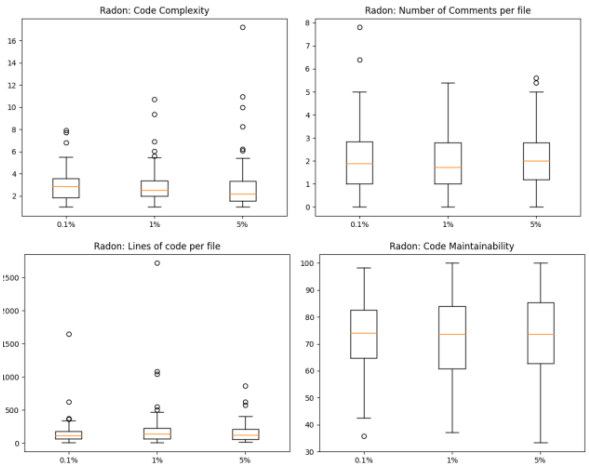


Figure 6: Box plots for the code complexity, comments per file, lines of code per file, and maintainability as measured by radon for users in the 0.1%, 1%, and 5% groups

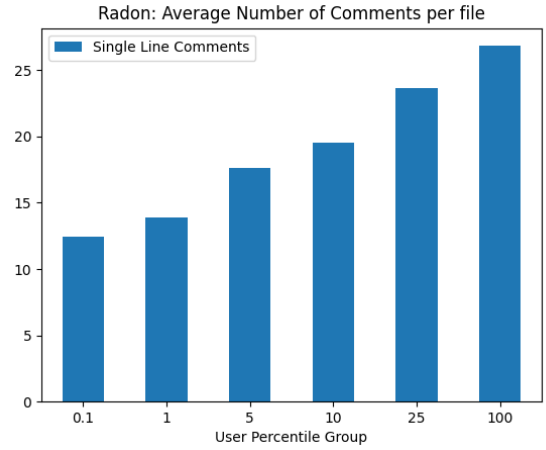


Figure 7: Histogram showing the number of comments per file for users across percentile groups from both the random and gold-star developer sets

7 Discussion

RQ 1: 10X engineers commit significantly more than others. Using novelty detection algorithms like Local Outlier Factor and Isolation Forest, along with analyzing commit frequency distributions using standard deviations, we confirm various tiers of users that have statistically extraordinary commit counts for Python projects. **RQ 2:** Our investigation regarding 10X engineers acting as contributors or maintainers yielded inconclusive results and would require further investigation. At first, we observed that while both groups favored being primary contributors of their own projects, 10X developers contributed significantly more to other projects than our randomly sampled developers did. However, after adjusting our random sampling procedure to require re-sampling and additional pre-processing, there was no longer a significant difference between the two groups. **RQ 3:** We found some indication that developers with lots of commits have less comments per file than others, but these results also require more investigation. More samples should be taken from the projects of each developer to give more conclusive results, and further comparison between groups is needed. **RQ 4:** Unlike earlier results, no significant tiers of developers emerged in these results. Surprisingly, it appears that the complexity, maintainability, and file length of projects that the top tiers of developers contribute to are relatively stable. Like RQ 3, further work should be done exploring more projects per user and comparing results across more percentiles to give conclusive results.

8 Threats to Validity

8.1 Internal

The users and projects we collected were not selected according to any stricter criteria than being individuals rather than organizations and python projects. Some additional filtration was applied to the generate Figures 7-9, which included avoiding projects with single-commits. The presence of these projects was infrequent and correlated with "dead accounts". It is possible that more rigorous pre-processing requirements for the data—such as requiring that the projects be of a certain age (as done in [10]) or filtering users more rigorously by minimum career contributions—could have an effect on our results. Additionally, when looking at the commits per user, we only looked at their commits to one randomly sampled project. This means we could be over or underestimating some users contributions if this random selection is a project that they are primary contributors to or almost never contribute to. Our hope is that our sample size is big enough to offset this, but it is a possible flaw in design.

One further caveat comes in how we measure the amount of commits from each user. We count commits over the lifetime of a user, which could privileges users who have longer histories. Due to constraints imposed by commit information available from the GitHub API and GHTorrent, we were also only able to collect the number of commits from each user rather than the amount of code in-

cluded in each commit. This could skew our results toward users who make lots of little commits but actually contribute less code than others (though whether or not either is correlated with 10X engineers is unknown).

8.2 External

As discussed in [4, 1], measuring productivity is a subjective thing. There is no one agreed metric for productivity, and there is disagreement regarding whether metrics such as amount of code produced or number of commits actually indicate a higher level of productivity. What we present here assumes that such measures are an accurate representation of programmer productivity.

9 Conclusion

It does appear that users exist who contribute significantly more to Python projects in terms of commits. If this is our metric of productivity, then the answer to the question of whether or not 10X engineers exist is yes. Developers exist with hundreds of thousands of commits more than others, and there seem to exist some distinct tiers that developers fall into. Using other metrics, such as pull requests, issue comments, and comments showed similar patterns. It remains an open question however whether the Python code written by these engineers has any major differences to that written by other engineers.

In future work, we hope to explore user commits in more detail, examining the size of each user commit in addition to the number contributed by each user. Something else we would like to examine is the amount of code reuse in 10X engineers' code (similar to work done in [10]). This could contribute to a better understanding of the programming practices of 10X engineers, especially when it comes to efficiency.

References

[1] C. Sürücü, B. Song, J. Krüger, G. Saake, and T. Leich, "Establishing key performance indicators for measuring software-development processes at a large organization," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 1331–1341. [Online]. Available: https://doi.org/10.1145/3368089.3417057

[2] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde, "What predicts software developers' productivity?" *IEEE Transactions on Software Engineering*, 2019. [Online]. Available: https://doi.org/10.1109/TSE.2019.2900308

[3] T. Z. Caitlin Sadowski, Ed., *Rethinking Productivity in Software Engineering*. Apress, Berkeley, CA, 2019. [Online]. Available: https://doi.org/10.1007/978-1-4842-4221-6

[4] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, "Software developers' perceptions of productivity," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 19–29. [Online]. Available: https://doi.org/10.1145/2635868.2635892

[5] W. Nichols, "The end to the myth of individual programmer productivity," *IEEE Software*, vol. 36, pp. 71–75, 09 2019.

[6] "Git awards." [Online]. Available: http://git-awards.com/

[7] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 233–236. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487132

[8] "Radon." [Online]. Available: https://pypi.python.org/pypi/radon.

[9] M. Halstead, "Elements of software science," 1977.

[10] M. Gharehyazie, B. Ray, M. Keshani, M. S. Zavosht, A. Heydarnoori, and V. Filkov, "Cross-project code clones in github," in *Emperical Software Engineering*, 2019, pp. 1538–1573.