REGULAR ARTICLE   **Open Access**

The penumbra of open source: projects outside of centralized platforms are longer maintained, more academic and more collaborative

Milo Z. Trujillo[1], Laurent Hébert-Dufresne[1,2] and James Bagrow[1,3]*

*Correspondence:
james.bagrow@uvm.edu
[1]Complex Systems Center,
University of Vermont, Burlington,
Vermont, USA
[3]Department of Mathematics &
Statistics, University of Vermont,
Burlington, Vermont, USA
Full list of author information is
available at the end of the article

Abstract

GitHub has become the central online platform for much of open source, hosting most open source code repositories. With this popularity, the public digital traces of GitHub are now a valuable means to study teamwork and collaboration. In many ways, however, GitHub is a convenience sample, and may not be representative of open source development off the platform. Here we develop a novel, extensive sample of public open source project repositories outside of centralized platforms. We characterized these projects along a number of dimensions, and compare to a time-matched sample of corresponding GitHub projects. Our sample projects tend to have more collaborators, are maintained for longer periods, and tend to be more focused on academic and scientific problems.

Keywords: Open source; Teamwork; Collective dynamics; Online social networks

1 Introduction

The GitHub hosting platform has long been recognized as a promising window into the complex world of online collaborations [10], open science [36], education [44], public sector work [31], and software development [21]. From 10 million repositories in 2014 [22], GitHub reported over 60 million new repositories in 2020 [14]. However, despite its size, there remain significant risks associated with GitHub as a data platform [23]. Without a baseline study examining open source development off of GitHub, it is unclear whether public GitHub data is a representative sample of software development practices or collaborative behavior. For studies of collaborations, it is particularly worrisome that most GitHub repositories are private and that most public ones are small and inactive [22]. These data biases have only grown in recent years as the platform stopped limiting the number of private repositories with fewer than four collaborators in 2019 [13].

Despite the fact that GitHub is not a transparent or unbiased window into collaborations, the popularity of the platform alone has proved very attractive for researchers. Early research focused on the value of transparency and working in public, analyzing how in-

dividuals select projects and collaborations [10], and conversely how collaborations grow and thrive [24, 32]. While fine-grained information about git commits within code repositories is readily available, higher-level findings about team collaboration and social software development practices are scarcer. Klug and Bagrow [24] introduce a metric of "effective team size," measuring each contributor's contributions against the distribution of GitHub events for the repository, distinguishing peripheral and "drive-by" contributors from more active team members. Choudhary et al. [6] focus on identifying "periods of activity" within a repository, beginning with a simple measurement of time dispersion between GitHub events, then identifying the participants and files edited in each burst of activity to determine productivity and partitioning of work according to apparent team dynamics.

Beyond looking at patterns of collaborations within projects, it is also useful to study GitHub as a social network, where collaborations are social ties mediated by repository [4, 28, 41]. These studies tend to offer results showing analogies between GitHub collaborations and more classic online social networks, such as modular structure [45] and heterogeneous distributions of collaborators per individual driven by rich-get-richer effects [4, 28]. More interestingly, studies also found that GitHub tends to show extremely low levels of reciprocity in actual social connections [28] and high levels of hierarchical, often star-like groups [45]. There are unfortunately few studies providing context for GitHub-specific findings, and no clear baseline to which they should be compared. Is GitHub more or less collaborative than other platforms of open source development? How much are collaborations shaped by the open source nature of the work, by the underlying technology, and by the platform itself? Altogether, it remains an open problem to quantify just how collaborative and social GitHub is.

GitHub is far from the only platform to host open source projects that use the Git version control system, but it is the most popular. What remains unclear is how much of the open source ecosystem now exists in GitHub's shadow, and how different these open source projects are when compared to their counterpart on the most popular public platforms. To this end, here we aim to study what we call the *Penumbra* of open source: Public repositories on public hosts other than the large centralized platforms (e.g. GitHub, GitLab, Sourceforge and other forges). Specifically, we want to compare the size, the nature and the temporal patterns of collaborations that occur in the Penumbra with that of a comparable random subset of GitHub.

Open source has long been linked to academic institutions [25], including libraries [5, 34], research centers [33, 35], and the classroom [43]. Version control systems such as git have been interesting tools to assist in classroom learning [7, 18], including computer science [11, 26] and statistics [2] courses. GitHub has played a role in the classroom and for hosting scientific research [12, 44], yet we expect many institutions to be either unwilling or unable to utilize GitHub or other commercial tools [8, 27, 43]. We therefore wish in this work to distinguish between academic and non-academic Penumbra hosts, in order to measure the extent with which academic institutions appear within the Penumbra ecosystem.

The rest of this paper is organized as follows. In Sect. 2 we describe our materials and methods, how we identify and collect Penumbra projects, how we gather a time-matched sample of GitHub projects, and we describe the subsequent analyses we perform on collected projects and the statistical models we employ. We report our results in Sect. 3 in-

cluding our analysis of our Penumbra sample and our comparison to our GitHub sample. Section 4 concludes with a discussion of our results, limitations of our study, and avenues for future work.

2 Materials and methods
2.1 Data collection
We began by identifying various open source software packages that can serve as self-hosted alternatives to GitHub. These included GitLab Community Edition (CE), Gitea, Gogs, cgit, RhodeCode, and SourceHut. We limited ourselves to platforms with a web-git interface similar to mainstream centralized platforms like GitHub and GitLab, and so chose to exclude command-line only source code management like GitoLite, as well as more general project management software like Jitsi and Phabricator. For each software package, we identified a snippet of HTML from each package's web interface that uniquely identifies that software. Often this was a version string or header, such as `<meta content="GitLab" property="og:site_name">`.

We then turned to Shodan [30] to find hosts running instances of each software package. Shodan maintains a verbose port scan of the entire IPv4 and some of the IPv6 Internet, including response information from each port, such as the HTML returned by each web server. This port scan is searchable, allowing us to list all web servers open to the public Internet that responded with our unique identifier HTML snippets. Notably, Shodan scans only include the default web page from each host, so if a web server hosts multiple websites and returns different content depending on the host in the HTTP request, then we will miss all but the front page of the default website. Therefore, Shodan results should be considered a strict under-count of public instances of these software packages. However, we have no reason to believe that it is a biased sample, as there are trade-offs to dedicated and shared web hosting for organizations of many sizes and purposes.

We narrowed our study to the three software packages with the largest number of public instances: GitLab CE, Gogs, and Gitea. Searching Shodan, we found 59,596 unique hosts. We wrote a web crawler for each software package, which would attempt to list every repository on each host, and would report when instances were unreachable (11,677), had no public repositories (44,863), or required login information to view repositories (2101). We then attempted to clone all public repositories, again logging when a repository failed to clone, sent us a redirect when we tried to clone, or required login information to clone. For each successfully cloned repository, we checked the first commit hash against the GitHub API, and set aside repositories that matched GitHub content (see Sect. 2.4). We discarded all empty (zero-commit) repositories. This left us with 45,349 repositories from 1558 distinct hosts.

Next, we wanted to collect a sample of GitHub repositories to compare development practices. We wanted a sample of a similar number of repositories from a similar date range, to account for trends in software development and other variation over time. We chose not to control for other repository attributes, like predominant programming language, size of codebase or contributorship, or repository purpose. We believe these attributes may be considered factors when developers choose where to host their code, so controlling for them would inappropriately constrain our analysis. To gather this comparison sample, we drew from GitHub Archive [17] via their BigQuery interface to find an equal number of "repository creation" events from each month a Penumbra repository

was created in. We attempted to clone each repository, but found that some repositories had since been deleted, renamed, or made private. To compensate, we oversampled from GitHub Archive for each month by a factor of 1.5. After data collection and filtering we were left with a time-matched sample of 57,914 GitHub repositories.

Lastly, to help identify academic hosts, we used a publicly available list of university domains.[1] This is a community-curated list, and so may contain geographic bias, but was the most complete set of university domains we located.

2.2 Host analysis

We used geoip lookups[2] to estimate the geographic distribution of hosts found in our Penumbra scan. We also created a simple labelling process to ascertain how many hosts were universities or research labs: Extract all unique emails from commits in each repository, and label each email as academic if the hostname in the email appears in our university domain list. If over 50% of unique email addresses on a host are academic, then the host is labeled as academic. This cutoff was established experimentally after viewing the distribution of academic email percentages per host, shown in the inset of Fig. 1(c). Under this cutoff, 15% of Penumbra hosts (130) were tagged as academic.

2.3 Repository analysis

We are interested in diverging software development practices between GitHub and the Penumbra, and so we measured a variety of attributes for each repository. To analyze the large number of commits in our dataset, we modified git2net [15] and PyDriller [38] to extract only commit metadata, ignoring the contents of binary "diff" blobs for performance. We measured the number of git branches per repository (later, in Fig. 2, we count only remote branches, and ignore origin/HEAD, which is an alias to the default branch), but otherwise concerned ourselves only with content in the main branch, so as to disambiguate measurements like "number of commits."

From the full commit history of the main branch we gather the total number of commits, the hash and time of each commit, the length in characters of each commit message, and the number of repository contributors denoted by unique author email addresses. (Email addresses are not an ideal proxy for contributors; a single contributor may use multiple email addresses, for example if they have two computers that are configured differently. Unfortunately, git commit data does not disambiguate usernames. Past work [16, 42] has attempted to disambiguate authors based on a combination of their commit names and commit email addresses, but we considered this out of scope for our work. By not applying identity disambiguation to either the Penumbra or GitHub repositories, the use of emails-as-proxy is consistent across both samples. If identity disambiguation would add bias, for example if disambiguation is more successful on formulaic university email addresses found on academic Penumbra hosts than it is on GitHub data, then using emails as identifiers will provide a more consistent view.) From the current state (head commit of the main branch) of the repository we measure the number of files per repository. This avoids ambiguity where files may have been renamed, split, or deleted in the commit his-

---

[1]https://github.com/hipo/university-domains-list

[2]https://dev.maxmind.com/geoip/geolite2-free-geolocation-data/

tory. We apply *cloc*,[3] the "Count Lines of Code" utility, to identify the top programming language per repository by file counts and by lines of code.

We also calculate several derived statistics. The average *interevent time*, the average number of seconds between commits per repository, serves as a crude indicator of how regularly contributions are made. We refine this metric as *burstiness*, a measure of the index of dispersion (or Fano Factor) of commit times in a repository [6]. The index of dispersion is defined as σ_w^2/μ_w, or the variance over the mean of events over some time window w. Previous work defines "events" broadly to encompass all GitHub activity, such as commits, issues, and pull requests. To consistently compare between platforms, we define "events" more narrowly as "commits per day". Note that while interevent time is only defined for repositories with at least two commits, burstiness is defined as 0 for single-commit repositories.

We infer the age of each repository as the amount of time between the first and most recent commit. One could compare the start or end dates of repositories using the first and last commit as well, but because we sampled GitHub by finding repositories with the same starting months as our Penumbra repositories, these measurements are less meaningful within the context of our study.

Following Klug and Bagrow [24], we compute three measures for how work is distributed across members of a team. The first, *lead workload*, is the fraction of commits performed by the "lead" or heaviest contributor to the repository. Next, a repository is *dominated* if the lead makes more commits than all other contributors combined (over 50% of commits). Note that all single-contributor repositories are implicitly dominated by that single user, and all two-contributor repositories are dominated unless both contributors have an exactly equal number of commits, so dominance is most meaningful with three or more contributors. Lastly, we calculate an *effective team size*, estimating what the effective number of team members would be if all members contributed equally. Effective team size m for a repository with M contributors is defined as $m = 2^h$, where $h = -\sum_{i=1}^{M} f_i \log_2 f_i$, and $f_i = w_i/W$ is the fraction of work conducted by contributors i. For example, a team with $M = 2$ members who contribute equally ($f_1 = f_2$) would also have an effective team size of $m = 2$, whereas a duo where one team member contributes 10 times more than the other would have an "effective" team size of $m = 1.356$. Effective team size is functionally equivalent to the Shannon entropy h, a popular index of diversity, but is exponentiated so values are reported in numbers of team members as opposed to the units of h, which are typically bits or nats. Since we only consider commits as work (lacking access to more holistic data on bug tracking, project management, and other non-code contributions [3]), f_i is equal to the fraction of commits in a repository made by a particular contributor. Interpreting the contents of commits to determine the magnitude of each contribution (as in expertise-detection studies like [9]) would add nuance, but would require building parsers for each programming language in our dataset, and requires assigning a subjective value for different kinds of contributions, and so is out of scope for our study. Therefore, the effective team size metric improves on a naive count of contributors, which would consider each contributor as equal even when their numbers of contributions differ greatly.

---

[3]https://github.com/AlDanial/cloc

2.4 Duplication and divergence of repositories

It is possible for a repository to be an exact copy or "mirror" of another repository and this mirroring may happen across datasets: a Penumbra repository could be mirrored on GitHub, for example. Quantifying the extent of mirroring is important for determining whether the Penumbra is a novel collection of open source code or if it mostly already captured within, for instance, GitHub. Likewise, a repository may have been a mirror at one point in the past but subsequent edits have caused one mirror to diverge from the other.

Searching for git commit hashes provides a reliable way to detect duplicate repositories, as hashes are derived from the cumulative repository contents[4] and, barring intentional attack [39] on older versions, hash collisions are rare. To determine the novelty of Penumbra repositories, we searched for their commit hashes on GitHub, on Software Heritage (SH), a large-scale archive of open source code [1] and within the Penumbra sample itself to determine the extent of mirroring within the Penumbra. Search APIs were used for GitHub and SH, while the Penumbra sample was searched locally. For each Penumbra repository, we searched for the first hash and, if the repository had more than one commit, the latest hash. If both hashes are found at least once on GitHub or SH, then we have a complete copy (at the time of data collection). If the first hash is found but not the second, then we know a mirror exists but has since diverged. If nothing is found, it is reasonable to conclude the Penumbra project is novel (i.e., independent of GitHub and SH).

To ensure a clean margin when comparing the Penumbra and GitHub samples, we excluded from our analysis (Sect. 2.3) any Penumbra repositories that were duplicated on GitHub, even if those duplicates diverged.

2.5 Statistical models

To understand better what features most delineate Penumbra and GitHub projects, we employ two statistical models: logistic regression and a random forest ensemble classifier. While both can in principle be used to predict whether a project belongs to the Penumbra or not, our goal here is inference: we wish to understand what features are most distinct between the two groups.

For logistic regression, we fitted two models. Exogenous variables were numbers of files, contributors, commits, and branches; average commit message length; average editors per file; average interevent time, in hours; lead workload, the proportion of commits made by the heaviest contributor; effective team size; burstiness, as measured by the index of dispersion; and, for model 1 only, the top programming language as a categorical variable. Given differences in programming language choice in academic and industry [37], we wish to investigate any differences when comparing Penumbra and GitHub projects (see also Sects. 2.1 and 3.3). There is a long tail of uncommon languages that prevents convergence when fitting model 1, so we processed the categorical variable by combining Bourne and Bourne Again languages and grouping languages that appeared in fewer than 1000 unique repositories into an "other" category before dummy coding. JavaScript, the most common language, was chosen as the baseline category. Missing values were present, due primarily to a missing top language categorization and/or an undefined average interevent time.

---

[4]Commit hashes include the files changed by the commit, and the hash of the parent commit, referencing a list of changes all the way to the start of the repository.

Empty or mostly empty repositories, as well as repositories with a single commit, will cause these issues, so we performed listwise deletion on the original data, removing repositories from our analysis when any fields were missing. After processing, we were left with 67,893 repositories (47.26% Penumbra). Logistic models were fitted using Newton-Raphson and odds e^{β} and 95% CI on odds were reported.
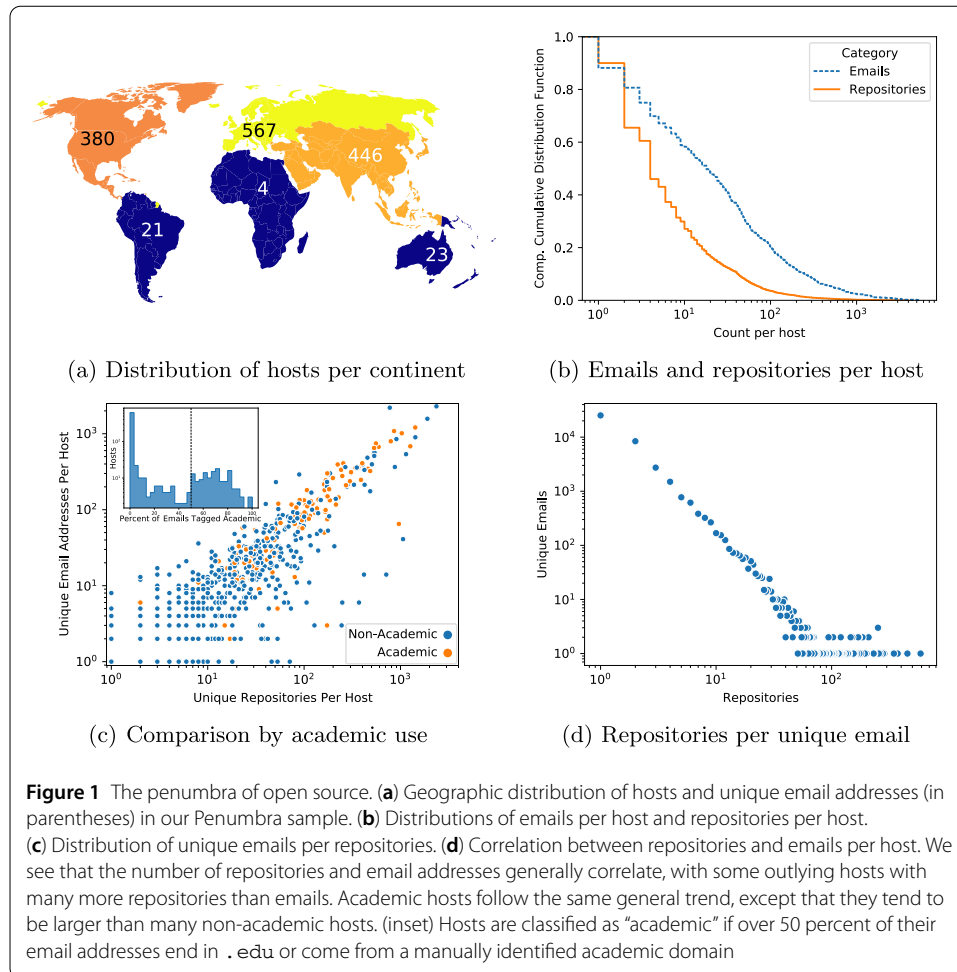
For the random forest model, feature importances were used to infer which features were most used by the model to distinguish between the two groups. We used the same data as logistic regression model 2, randomly divided into 90% training, 10% validation subsets. We fit an ensemble of 1000 trees to the training data using default hyperparameters; random forests were fit using scikit-learn v0.24.2. Model performance was assessed using an ROC curve on the validation set. Feature importances were measured with permutation importance, a computationally-expensive measure of importance but one that is not biased in favor of features with many unique values [40]. Permutation importance was computed by measuring the fitted model's accuracy on the validation set; then, the values of a given feature were permuted uniformly at random between validation observations and validation accuracy was recomputed. The more accuracy drops, the more important that feature was. Permutations were repeated 100 times per feature and the average drop in accuracy was reported. Note that permutation importance may be negative for marginally important features and that importance is only useful as a relative quantity for ranking features within a single trained (ensemble) model.

3 Results

We sampled the Penumbra of the open-source ecosystem: Public repositories on public hosts independent from large centralized platforms. Our objective is to compare the Penumbra to GitHub, the largest centralized platform, to better understand the representativeness of GitHub as a sample of the open-source ecosystem and how the choice of platforms might influence online collaborations. In Sect. 3.1 we begin with an overview of the Penumbra's geographic distribution and the scale of hosts. In Sect. 3.2 we analyze the collaboration patterns and temporal features of Penumbra and GitHub repositories. Section 3.3 examines the programming language domains of Penumbra and GitHub projects while Sect. 3.4 further investigates differences between academic and non-academic Penumbra repositories. Statistical models in Sect. 3.5 summarize the combined similarities and differences between Penumbra and GitHub repositories. Finally, in Sect. 3.6 we investigate the novelty of our Penumbra sample, how many Penumbra repositories are duplicates and whether Penumbra repositories also exist on GitHub and within the Software Heritage [1] archive.

3.1 An overview of the Penumbra sample

Our Penumbra sample consists of 1558 distinct hosts from all six inhabited continents and 45,349 non-empty repositories with no matching commits on GitHub (Sect. 2.4; we explore overlap with GitHub in Sect. 3.6). This geographic distribution, illustrated in Fig. 1 and described numerically in Table 1, shows that the Penumbra is predominantly active in Europe, North America, and Asia by raw number of hosts and repositories. However, Oceania has the second most repositories per capita, and the highest percentage of academic emails in commits from repositories cloned from those hosts (Table 1). Overall, the geographic spread of the Penumbra is similar to GitHub's self-reported distribution

Figure 1 The penumbra of open source. (**a**) Geographic distribution of hosts and unique email addresses (in parentheses) in our Penumbra sample. (**b**) Distributions of emails per host and repositories per host. (**c**) Distribution of unique emails per repositories. (**d**) Correlation between repositories and emails per host. We see that the number of repositories and email addresses generally correlate, with some outlying hosts with many more repositories than emails. Academic hosts follow the same general trend, except that they tend to be larger than many non-academic hosts. (inset) Hosts are classified as "academic" if over 50 percent of their email addresses end in `.edu` or come from a manually identified academic domain

Table 1 Geographic split of our Penumbra (PN) and GitHub (GH) [14] samples

Region	% Hosts	% PN users	% GH users [14]	PN repositories (per capita)	% Unique emails from academic domains
EU	39.35	73.47	26.8	83,612 (1.52×10^{-4})	39.20
NA	26.37	15.81	34	51,245 (2.97×10^{-4})	41.22
AS	30.95	7.38	30.7	21,765 (1.55×10^{-5})	1.21
SA	1.46	1.36	4.9	2776 (1.64×10^{-5})	12.41
OC	1.60	1.93	1.7	3347 (2.58×10^{-4})	65.24
AF	0.28	0.04	2	215 (9.44×10^{-7})	0.00

of users [14], but with a stronger European emphasis and even less Southern Hemisphere representation.

We find a strong academic presence in the Penumbra: on 15% of hosts, more than half of email addresses found in commits come from academic domains (see also Sect. 3.4). These academic hosts make up many of the larger hosts, but represent a minority of all Penumbra repositories (37% of non-GitHub-mirrors). We plotted the "size" of each host in terms of unique emails and repositories, as well as its academic status, in Fig. 1(c). We find that while academic hosts tend not to be "small", they do not dominate the Penumbra in terms of user or repository count, refuting the hypothesis that most Penumbra activity is academic.
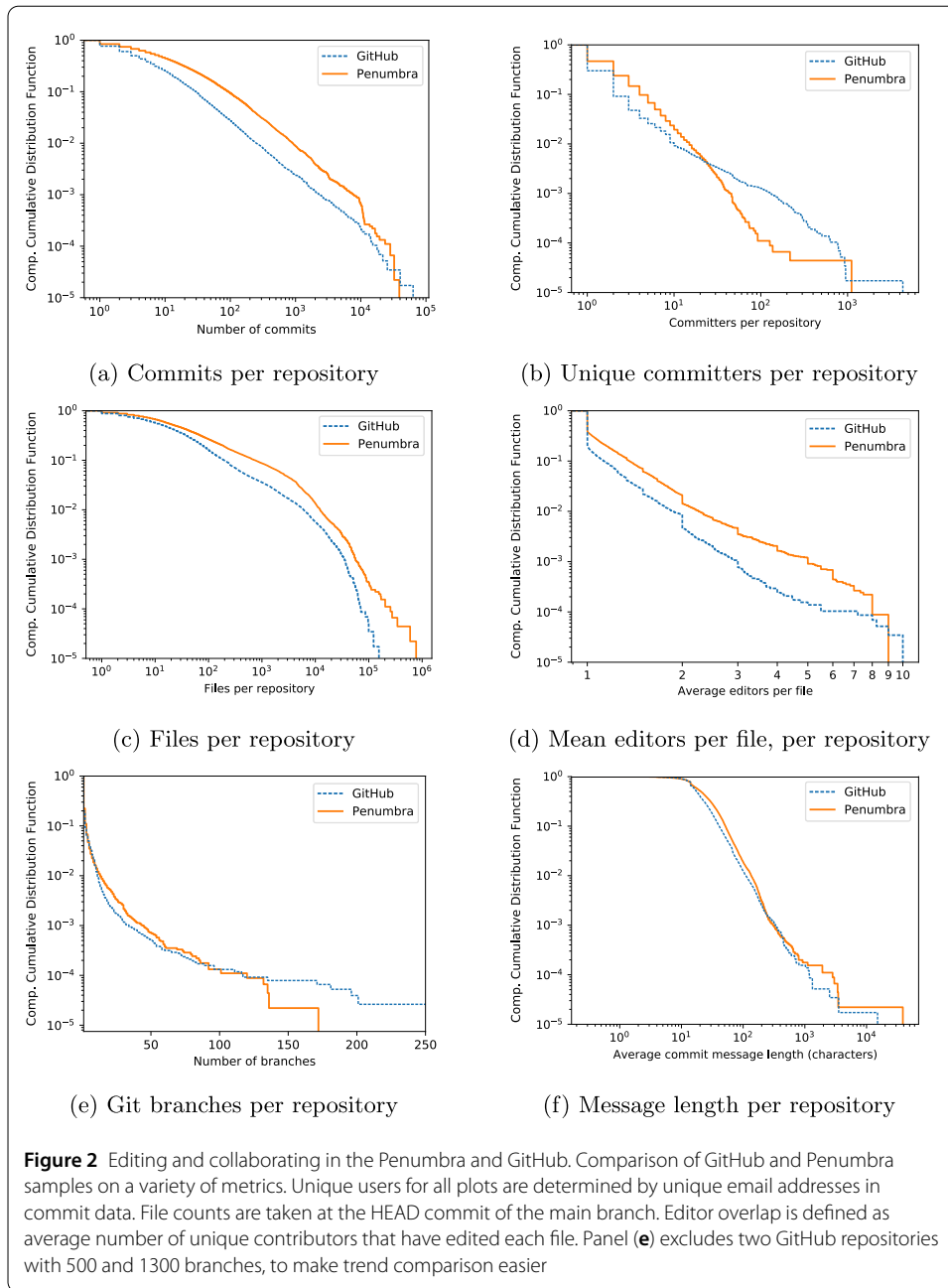
We are also interested in how distinct hosts are: How many repositories do users work on, and are those repositories all on a single host, or do users contribute to code on multiple hosts? To investigate, we first plot the number of unique email addresses per host in Fig. 1(b), then count the number of email addresses that appear on multiple hosts. Critically, users may set a different email address on different hosts (or even unique emails per-repository, or per-commit, although this would be tedious and unlikely), so using email addresses as a proxy for "shared users" offers only a lower-bound on collaboration. We find that 91.7% of email addresses in our dataset occur on only one host, leaving 3435 email addresses present on 2-4 hosts. Fifteen addresses appear on 5-74 hosts, but all appear to be illegitimate, such as "you@example.com", emails without domain suffixes like "admin" or "root@localhost", and a few automated systems like "anonymous@overleaf.com". We find 61 email addresses on hosts in two or more countries (after removing fake email addresses by the aforementioned criteria), and 33 on multiple continents (after the same filtering).

We did not repeat this analysis on our GitHub sample, because the dataset is too different for such a comparison to be meaningful. All GitHub repositories are on a single "host", so there is no analogue to "multi-host email addresses". We considered comparing distributions of "repositories committed to by each email", but ruled this out because of our data collection methodology. For each Penumbra host, we have data on every commit in every public repository, giving us a complete view of each user's contributions. For GitHub however, we have a small sample of repositories from the entire platform, so we are likely to miss repositories that each GitHub user contributed to.

3.2 Collaboration patterns and temporal features

We compare software development and collaboration patterns between our Penumbra sample and a GitHub sample of equivalent size and time period (Fig. 2 and Table 2). We examine commits per repository, unique emails per repository (as a proxy for unique contributors), files per repository, average editors per file, branches per repository, and commit message length. While mean behavior was similar in both repository samples, diverging tail distributions show that Penumbra repositories usually have more commits, more files, fewer emails, and more editors per file.

One might hypothesize that with more files and fewer editors the Penumbra would have stronger "partitioning", with each editor working on a different subset of files. However, our last three metrics suggest that the Penumbra has more collaborative tendencies: while Penumbra repositories are larger (in terms of files), with smaller teams (in terms of editors), there are on average *more* contributors working on the same files or parts of a project. To deepen our understanding of this collaborative behavior, we also estimated the "effective team size" for each repository by the fraction of commits made by each editor. This distinguishes consistent contributors from editors with only a handful of commits, such as "drive-by committers" that make one pull request, improving upon a naive count of unique emails. These estimates show that while there are more GitHub repositories with one active contributor, and more enormous projects with over 50 team members, the Penumbra has more repositories with between 2 and 50 team members. However, for all team sizes between 2 and 10, we find that more penumbra repositories are "dominated" by a single contributor, see Fig. 3(f), meaning that their top contributor has made over 50% of all commits.

Figure 2 Editing and collaborating in the Penumbra and GitHub. Comparison of GitHub and Penumbra samples on a variety of metrics. Unique users for all plots are determined by unique email addresses in commit data. File counts are taken at the HEAD commit of the main branch. Editor overlap is defined as average number of unique contributors that have edited each file. Panel (**e**) excludes two GitHub repositories with 500 and 1300 branches, to make trend comparison easier

We also compare temporal aspects of Penumbra and GitHub repositories (Fig. 3). Penumbra repositories are shown to be generally older in terms of "time between the first and most recent commit" in Fig. 3(a), have more commits in Fig. 3(b), but are also shown to have a longer time between commits measured both as interevent time in Fig. 3(c), and as burstiness in Fig. 3(d). This means that while Penumbra repositories are maintained for longer (or conversely, there are many short-lived repositories on GitHub that receive no updates), they are maintained inconsistently or in a bursty pattern, receiving updates after long periods of absence. And while both GitHub and Penumbra repositories tend to be bursty, a larger portion of Penumbra repositories exhibit burstiness as indicated by an index of dispersion above 1.

Table 2 Comparison of Penumbra and GitHub datasets

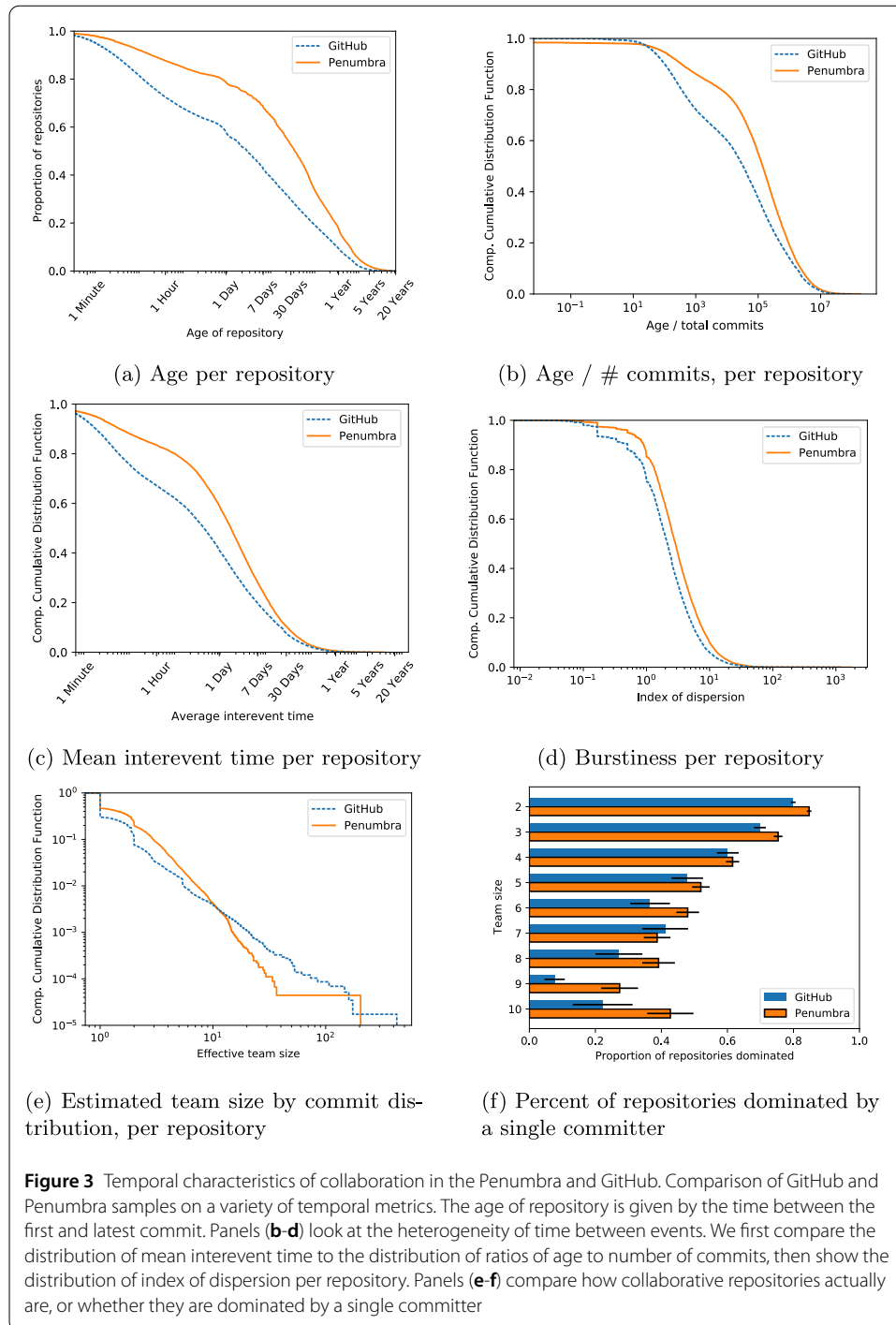| Statistic	Fig.	Penumbra			GitHub			KS 2-Sample	
		Mean	Median	CI	Mean	Median	CI	KS S	KS P
Files	2(c)	244.47	12	[1, 859]	156.07	9	[1, 264]	0.07	<0.001
Committers	2(b)	2.39	1	[1, 6]	2.08	1	[1, 3]	0.17	<0.001
Message Lengths	2(f)	29.24	20.80	[7.00, 67.33]	24.23	17.60	[7.42, 56.00]	0.13	<0.001
Editor Density	2(d)	1.12	1.00	[1.00, 1.60]	1.05	1.00	[1.00, 1.30]	0.20	<0.001
Burstiness	3(d)	4.86	2.88	[0.50, 14.51]	3.68	2.15	[0.17, 11.24]	0.13	<0.001
Commits	2(a)	67.12	8	[1, 194]	25.27	4	[1, 57]	0.20	<0.001
Branches	2(e)	1.74	1	[1, 4]	1.67	1	[1, 5]	0.03	<0.001
Age (hours)	3(a)	5528	883	[0.1, 25556]	2669	73	[0.03, 16194]	0.26	<0.001
Age / Commits	3(b)	283	39	[0.02, 1261]	193	9	[0.01, 944]	0.19	<0.001
Avg. Interevent	3(c)	375	43	[0.05, 1547]	257	11	[0.02, 1130]	0.19	<0.001
Team Size	3(e)	1.71	1.00	[1.00, 3.92]	1.42	1.00	[1.00, 2.67]	0.17	<0.001

Mean, median, and 5th and 95th percentile values from the Penumbra and GitHub samples for each statistic. *KS S* and *KS P* represent the Kolmogorov-Smirnov two-sample statistic, and its corresponding p-value.

3.3 Language domains

Most of our analysis has focused on repository metadata (commits and files), rather than the content of the repositories. This is because more in-depth content comparison, such as the dependencies used, or functions written within a repository's code, would vary widely between languages and require complex per-language parsing. However, we have classified language prevalence across the Penumbra and GitHub by lines of code and file count per repository in Fig. 4(left column). We find that the Penumbra emphasizes academic languages (TeX) and older languages (C, C++, PHP, Python), while GitHub represents more web development (JavaScript, TypeScript, Ruby), and mobile app development (Swift, Kotlin, Java). We additionally compare repositories within the Penumbra that come from academic hosts (> 50% emails come from academic domains; see Methods) and non-academic hosts, using the same lines of code and file count metrics in Fig. 4(right column). Academic hosts unsurprisingly contain more languages used in research (Python, MAT-LAB, and Jupyter notebooks), and languages used in teaching (Haskell, assembly, C). Despite Java's prevalence in enterprise and mobile app development, and JavaScript's use in web development, academic hosts also represent more Java and Typescript development. By contrast, non-academic hosts contain more desktop or mobile app development (Objective C, C#, QT), web development (JavaScript, PHP), shell scripts and docker files, and, surprisingly, Pascal.

3.4 Academic and non-academic hosts

Academic hosts account for over 15% of hosts and 37% of repositories in the Penumbra, so one might hypothesize that academic software development has a striking effect on the differences between GitHub and the Penumbra. To investigate this, Fig. 5 redraws Figs. 2(d) and 3(a) with academic and non-academic Penumbra repositories distinguished. We find that the academic repositories are maintained for about the same length of time as their non-academic counterparts, and that academic repositories have fewer editors per file than non-academic development. In fact, academic repositories more closely match GitHub repositories in terms of editors per file. Therefore, we find that academic software development does not drive the majority of the differences between GitHub and the Penumbra.

(a) Age per repository

(b) Age / # commits, per repository

(c) Mean interevent time per repository

(d) Burstiness per repository

(e) Estimated team size by commit distribution, per repository

(f) Percent of repositories dominated by a single committer
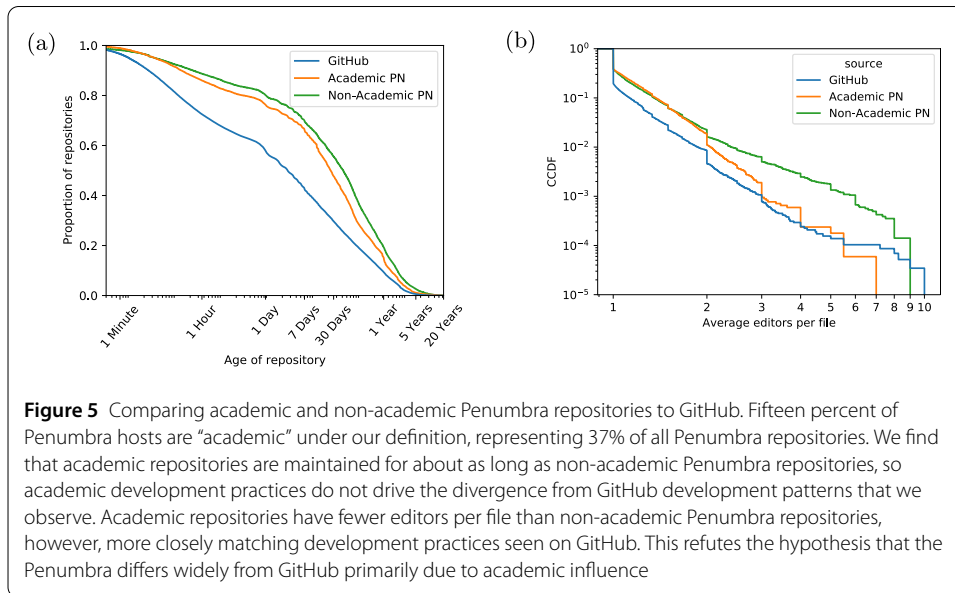
Figure 3 Temporal characteristics of collaboration in the Penumbra and GitHub. Comparison of GitHub and Penumbra samples on a variety of temporal metrics. The age of repository is given by the time between the first and latest commit. Panels (**b**-**d**) look at the heterogeneity of time between events. We first compare the distribution of mean interevent time to the distribution of ratios of age to number of commits, then show the distribution of index of dispersion per repository. Panels (**e**-**f**) compare how collaborative repositories actually are, or whether they are dominated by a single committer

3.5 Statistical models

To understand holistically how these different features delineate the two data sources, we perform combined statistical modeling. First, we performed logistic regression (Table 3) on the outcome variable of GitHub vs. Penumbra, see Sect. 2.5 for details. We fit two models, one containing the primary programming language as a feature and the other not. Examining the odds e^{β} for each variable, we can determine which variables, with other variables held constant, most clearly distinguish GitHub and Penumbra repositories.

Figure 4 Dominant language domains in the Penumbra and GitHub. Comparison of language popularity, measured by lines of code (LOC) in panels (**a**-**b**) and by file count in panels (**c**-**d**). We count the top languages of each repository by the specified metric, normalize the results to a percentage of independent or GitHub repositories, then subtract the percentages. Therefore a language with a value of –0.05, for example, is the top language on 5% more GitHub repositories than Penumbra repositories, while a positive value indicates 5% more Penumbra repositories than GitHub repositories

The strongest non-language separators are average editors per file, lead workload, and the number of contributors. The strongest language separators are TeX, C/C++ Headers, and C++. The odds on these variables underscore our existing results: Penumbra projects have more editors per file and less workload placed upon the lead contributor. Likewise, the odds on TeX and C/C++ code make it more likely for Penumbra projects to be focused on academic and scientific problems.

Supplementing our logistic models we also used nonlinear random forest regressions trained to predict whether a project was in GitHub or the Penumbra. While trained models can be used as predictive classifiers, our goal is to interpret which model features are used to make those predictions, so we report in Fig. 6 the top-ten feature importances (Sect. 2.5) in our model. Here we find some differences and similarities with the (linear) logistic regression results. Both average editors per file and number of contributors were important, but the random forest found that lead workload was not particularly important. However, the most important features for the random forests were average interevent time, burstiness, and number of commits. (All three were also significant in the logistic regression models.) The overall predictive performance of the random forest is reasonable (Fig. 6 inset). Taken together, the random forest is especially able to separate the two classes of projects based on time dynamics.

Figure 5 Comparing academic and non-academic Penumbra repositories to GitHub. Fifteen percent of Penumbra hosts are "academic" under our definition, representing 37% of all Penumbra repositories. We find that academic repositories are maintained for about as long as non-academic Penumbra repositories, so academic development practices do not drive the divergence from GitHub development patterns that we observe. Academic repositories have fewer editors per file than non-academic Penumbra repositories, however, more closely matching development practices seen on GitHub. This refutes the hypothesis that the Penumbra differs widely from GitHub primarily due to academic influence

Table 3 Logistic regression models for GitHub vs. Penumbra outcome

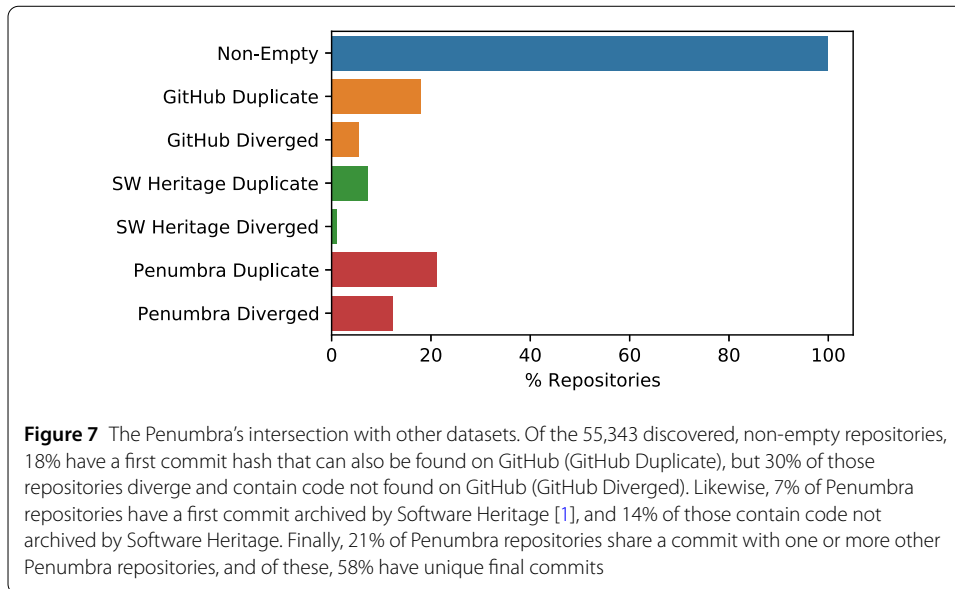|	Model 1			Model 2		
	e^β	p	CI	e^β	p	CI
Constant	0.188	<0.001	[0.156, 0.225]	0.435	<0.001	[0.364, 0.520]
Language (vs. JavaScript)						
Bourne (Again) Shell	3.478	<0.001	[3.162, 3.826]			
C	4.065	<0.001	[3.671, 4.502]			
C#	1.589	<0.001	[1.444, 1.750]			
C++	5.636	<0.001	[5.184, 6.127]			
C/C++ Header	5.829	<0.001	[5.103, 6.657]			
Java	2.192	<0.001	[2.070, 2.321]			
Jupyter Notebook	2.722	<0.001	[2.459, 3.012]			
OTHER	2.124	<0.001	[2.023, 2.230]			
PHP	2.524	<0.001	[2.323, 2.743]			
Python	2.804	<0.001	[2.651, 2.965]			
TeX	30.641	<0.001	[25.348, 37.040]			
TypeScript	1.078	0.187	[0.964, 1.205]			
Vuejs Component	4.940	<0.001	[4.331, 5.635]			
Files	1.000	<0.001	[1.000, 1.000]	1.000	<0.001	[1.000, 1.000]
Commits	1.001	<0.001	[1.001, 1.001]	1.001	<0.001	[1.001, 1.001]
Average editors per file	3.337	<0.001	[3.002, 3.709]	3.328	<0.001	[3.002, 3.689]
Average message length	1.002	<0.001	[1.001, 1.002]	1.002	<0.001	[1.001, 1.003]
Burstiness	1.059	<0.001	[1.055, 1.063]	1.058	<0.001	[1.053, 1.062]
Average interevent time [h]	1.000	<0.001	[1.000, 1.000]	1.000	<0.001	[1.000, 1.000]
Branches	0.971	<0.001	[0.965, 0.976]	0.961	<0.001	[0.955, 0.966]
Lead workload	0.461	<0.001	[0.407, 0.522]	0.433	<0.001	[0.384, 0.489]
Committers	0.945	<0.001	[0.936, 0.955]	0.940	<0.001	[0.931, 0.950]
Effective team size	1.010	0.516	[0.980, 1.040]	1.016	0.315	[0.985, 1.047]
-2LL	84,478.844			89,788.631		
Pseudo-R2	0.100			0.044		
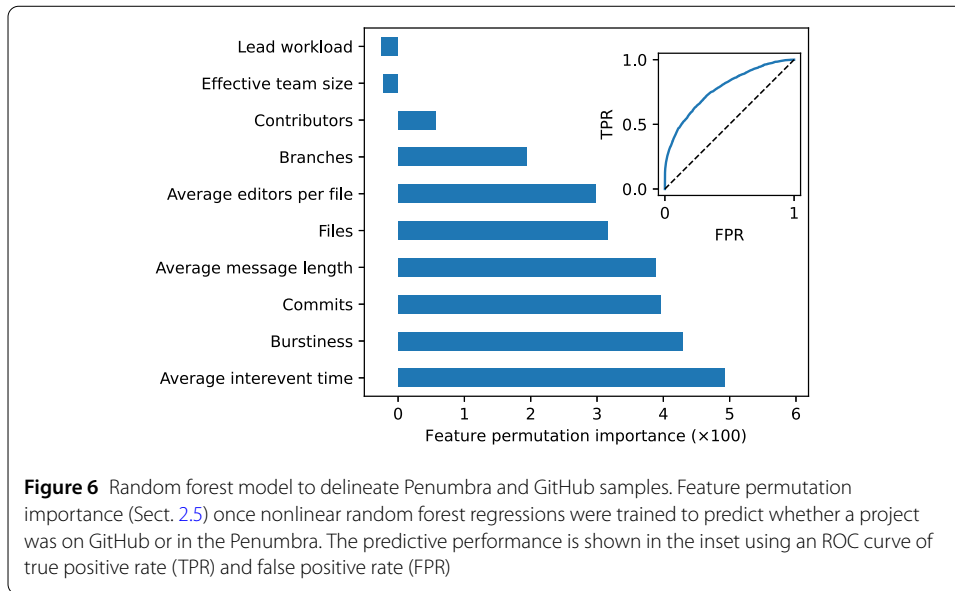
Figure 6 Random forest model to delineate Penumbra and GitHub samples. Feature permutation importance (Sect. 2.5) once nonlinear random forest regressions were trained to predict whether a project was on GitHub or in the Penumbra. The predictive performance is shown in the inset using an ROC curve of true positive rate (TPR) and false positive rate (FPR)

3.6 Novelty of the Penumbra sample

How novel are the repositories we have discovered in the Penumbra? It may be that many Penumbra repositories are "mirrored" on GitHub, in which case the collected Penumbra sample would not constitute especially novel data. In contrast, if few repositories appear on GitHub, then we can safely conclude that the Penumbra is a novel collection of open source code. To test the extent that the Penumbra is independent of GitHub, we checked the first commit hash of each Penumbra repository against the GitHub Search API (Sect. 2.4). We found 9994 such repositories (Fig. 7) and conclude that the majority of Penumbra repositories are novel. We excluded these overlapping repositories from our comparisons between the Penumbra and GitHub. However, such repositories may not represent true duplicates, but instead "forks", where developers clone software from GitHub to the Penumbra and then make local changes, or vice-versa, leading to diverging code. To disambiguate, we checked the *last* commit hash from each of the 9994 overlapping repositories against the GitHub API, and found 3056 diverging commits, as illustrated in Fig. 7. In other words, 30% of Penumbra repositories with a first commit on GitHub also contain code *not* found on GitHub. While we still excluded these repositories to ensure a wide margin between the samples, in fact, the differences in these repositories further underscore the novelty of the Penumbra data.

We also compared our repositories against Software Heritage [1], an archive of open source software development. While Software Heritage is not a hosting platform like GitHub, it represents a potentially similar dataset to our own. Applying the same methodology as for GitHub mirror detection, we found that 4053 repositories (9% of our non-empty Penumbra sample) had a matching first commit hash archived on Software Heritage, and that of these, 564 repositories (14% of overlapping first commits) contained code *not* archived by Software Heritage. Since Software Heritage is an archive, rather than a software development platform, we did not filter out the 4053 overlapping repositories from our comparisons between the Penumbra and GitHub. We again conclude that our Penumbra sample is primarily not captured by Software Heritage; see also Discussion.

Figure 7 The Penumbra's intersection with other datasets. Of the 55,343 discovered, non-empty repositories, 18% have a first commit hash that can also be found on GitHub (GitHub Duplicate), but 30% of those repositories diverge and contain code not found on GitHub (GitHub Diverged). Likewise, 7% of Penumbra repositories have a first commit archived by Software Heritage [1], and 14% of those contain code not archived by Software Heritage. Finally, 21% of Penumbra repositories share a commit with one or more other Penumbra repositories, and of these, 58% have unique final commits

We additionally looked for mirrors and forks within the Penumbra, shown in Fig. 7. As when comparing to external datasets, we found repositories that shared a first commit hash, then checked whether the last commit hash diverged. We find 11,717 Penumbra repositories share a first commit with at least one other, which constitutes 25.88% of non-empty Penumbra repositories. These mirrors come from a total of 3348 initial commits. Of these repositories, 6806 share a *last* commit with one or more repositories, suggesting that they have not diverged since creation. Notably, 1287 of the forks and mirrors contain only a single commit. Over a third of the forks and mirrors are on academic hosts (39.46%, 4623 repositories), which is especially notable because academic hosts constitute only 15% of our dataset. As a ratio, we find 35.56 mirrors per academic host, 9.98 per non-academic host. This would fit an educational use-case, such as a class assignment where each student clones an initial repository and then works independently.

4 Discussion

In this paper, we collected independent git hosts to sample what we call the *Penumbra* of the open source ecosystems: public hosts outside of the large, popular, centralized platforms like GitHub. Our objective was to compare a sample of the Penumbra to GitHub to evaluate the representativeness of GitHub as a data source and identify the potential impact of a platform on the work it hosts. In doing so, we found that projects outside of centralized platforms were more academic, longer maintained, and more collaborative than those on GitHub. These conclusions were obtained by looking at domains of email addresses of user accounts in the repositories, as well as measuring temporal and structural patterns of collaborations therein.

Importantly, projects in the Penumbra also appear to be more heterogeneous in important ways. Namely, we find more skewed distributions of files per repository and average number of editors per file, as well as more bursty patterns of editing. These bursty patterns are characterized by a skewed distribution of interevent time; meaning, projects in the Penumbra are more likely to feature long periods without edits before periods of rapid editing. Altogether, our results could suggest that the popularity and very public nature of

GitHub might contribute to a large amount of low-involvement contributors (or so-called "drive-by" contributions).

Our current sample of the Penumbra is extensive, but our methodology for identifying hosts presents shortcomings. Most notably, of the approximately 60,000 GitLab CE, Gitea, and Gogs instances identified by Shodan, only 13.4% provided public access to one or more repositories. We can say little about the hosts that provide no public access, and therefore constitute the dark shadow of software development. Further, Shodan may not capture all activity on a given server: it identifies hosts by their responses to a request for the front page, and is not a complete web crawler (Sect. 2.1). While this was sufficient in identifying 60,000 hosts, it is an under-estimate of the true number of Penumbra hosts, meaning that our dataset remains a sample of the full Penumbra and there exists room for improvement.

We determined from commit hashes that our sample of the Penumbra is mostly disjoint from GitHub and from the Software Heritage archive. This shows that our strategy of seeking public hosts using Shodan is a viable way to uncover novel sources of code. Archival efforts such as Software Heritage and World of Code [29] can benefit from this work as they can easily integrate our sampling method into their archiving process. Doing so can help them further achieve their goals of capturing as much open source software as possible.

There remain several open questions about our sample of the Penumbra worth further pursuit. For instance, the observed shift in languages used on Penumbra repositories implies that they tend to have more focus on academic and/or scientific projects than GitHub. However, programming language alone is a coarse signal of the intent and context of a given project. Future work should attempt a natural language analysis of repository contents to better identify the type of problems tackled in different regions of the open-source ecosystem. Furthermore, this would allow researchers to match Penumbra and GitHub repositories by the problem-spaces they address, indicating whether developers off of GitHub solve similar problems in different ways.

There are also several important demographic questions regarding, among others, the age, gender, and nationality of users in the Penumbra. GitHub is overwhelmingly popular in North America [14] and therefore does not provide uniform data on members of the open-source community. Critical new efforts could attempt to assess the WEIRDness — i.e., the focus on Western, educated, industrialized, rich and democratic populations [19, 20] — of GitHub as a convenience sample.

Digging further into the code or user demographics of the Penumbra would allow us to answer new questions about the interplay of code development with the platform that supports it. How does the distribution of developer experience levels affect projects, teams and communities? What are the key differences in intent, practices and products based on how open and public the platform is? Who contributes to the work and does it differ depending on the platform [3]?

We are only beginning to explore the space of open source beyond GitHub and other major central platforms. The Penumbra hosts explored here are fundamentally harder to sample and analyze. The hosts themselves have to be found and not all hosts provide public access. Unlike GitHub, we do not have a convenient API for sampling the digital traces of collaborations, so the underlying git repositories must be analyzed directly. There is therefore much of the open source ecosystem left to explore. Yet only by exploring new

regions, as we did here, can we fully understand how online collaborative work is affected by the platforms and technologies that support it.

Availability of data and materials
The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
MZT conceived of the presented idea, implemented and conducted data collection, and performed data analysis. JB implemented and applied statistical models. LHD and JB supervised the project and planned out data collection and analysis. All authors wrote the final manuscript. All authors read and approved the final manuscript.

Author details
[1]Complex Systems Center, University of Vermont, Burlington, Vermont, USA. [2]Department of Computer Science, University of Vermont, Burlington, Vermont, USA. [3]Department of Mathematics & Statistics, University of Vermont, Burlington, Vermont, USA.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Abramatic JF, Di Cosmo R, Zacchiroli S (2018) Building the universal archive of source code. Commun ACM 61(10):29–31
2. Beckman MD, Çetinkaya-Rundel M, Horton NJ, Rundel CW, Sullivan AJ, Tackett M (2021) Implementing version control with Git and Github as a learning objective in statistics and data science courses. J Stat Data Sci Educ. 29(sup1):S132–S144
3. Casari A, McLaughlin K, Trujillo MZ, Young JG, Bagrow JP, Hébert-Dufresne L (2021) Open source ecosystems need equitable credit across contributions. Nat Comput Sci 1(1):2
4. Celińska D (2018) Coding together in a social network: collaboration among GitHub users. In: Proceedings of the 9th international conference on social media and society, pp 31–40
5. Chen HL, Zhang Y (2014) Functionality analysis of an open source repository system: current practices and implications. J Acad Librariansh 40(6):558–564
6. Choudhary SS, Bogart C, Rosé CP, Herbsleb JD (2018) Modeling coordination and productivity in open-source GitHub projects. Carnegie-Mellon Univ Inst of Software Research International, Tech Rep pp CMU–ISR–18–101
7. Clifton C, Kaczmarczyk LC, Mrozek M (2007) Subverting the fundamentals sequence: using version control to enhance course management. SIGCSE Bull 39(1):86–90
8. Coll H, Bri D, Garcia M, Lloret J (2008) Free software and open source applications in higher education. In: WSEAS international conference. Proceedings. Mathematics and computers in science and engineering, WSEAS, 5
9. da Silva JR, Clua E, Murta L, Sarma A (2015) Niche vs. breadth: calculating expertise over time through a fine-grained analysis. In: 2015 IEEE 22nd international conference on software analysis, evolution, and reengineering (SANER). IEEE, pp 409–418
10. Dabbish L, Stuart C, Tsay J, Herbsleb J (2012) Social coding in GitHub: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 conference on computer supported cooperative work, pp 1277–1286
11. Dorodchi M, Dehbozorgi N (2016) Utilizing open source software in teaching practice-based software engineering courses. In: 2016 IEEE frontiers in education conference (FIE), pp 1–5
12. Feliciano J, Storey MA, Zagalsky A (2016) Student experiences using Github in software engineering courses: a case study. In: 2016 IEEE/ACM 38th international conference on software engineering companion (ICSE-C). IEEE, pp 422–431
13. GitHub (2019) New year, new GitHub: announcing unlimited free private repos and unified Enterprise offering. https://github.blog/2019-01-07-new-year-new-github/, accessed: 2021-06-14
14. GitHub (2020) The 2020 state of the octoverse. https://octoverse.github.com/, accessed: 2021-06-14

15. Gote C, Scholtes I, Schweitzer F (2019) Git2net: mining time-stamped co-editing networks from large git repositories. In: Proceedings of the 16th international conference on mining software repositories. IEEE Press, New York, pp 433–444
16. Gote C, Zingg C (2021) Gambit–an open source name disambiguation tool for version control systems. In: Proceedings of the 18th international conference on mining software repositories
17. Grigorik I (2012) The GitHub archive. https://githubarchive.org
18. Haaranen L, Lehtinen T (2015) Teaching git on the side: version control system as a course platform. In: Proceedings of the 2015 ACM conference on innovation and technology in computer science education, pp 87–92
19. Henrich J, Heine SJ, Norenzayan A (2010) Beyond WEIRD: towards a broad-based behavioral science. Behav Brain Sci 33(2–3):111
20. Henrich J, Heine SJ, Norenzayan A (2010) Most people are not weird. Nature 466(7302):29
21. Kalliamvakou E, Damian D, Blincoe K, Singer L, German DM (2015) Open source-style collaborative development practices in commercial projects using GitHub. In: 2015 IEEE/ACM 37th IEEE international conference on software engineering, IEEE, vol 1, pp 574–585
22. Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D (2014) The promises and perils of mining GitHub. In: Proceedings of the 11th working conference on mining software repositories, pp 92–101
23. Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D (2016) An in-depth study of the promises and perils of mining GitHub. Empir Softw Eng 21(5):2035–2071
24. Klug M, Bagrow JP (2016) Understanding the group dynamics and success of teams. R Soc Open Sci 160:007
25. Lakhani KR, Wolf RG (2005) Why hackers do what they do: understanding motivation and effort in free/open source software projects. In: Feller J, FitzGerald B, Hissam S, Lakhani K (eds) Perspectives on free and open source software. MIT Press, Cambridge
26. Lawrance J, Jung S, Wiseman C (2013) Git on the cloud in the classroom. In: Proceeding of the 44th ACM technical symposium on computer science education. SIGCSE '13. Association for Computing Machinery, New York, pp 639–644
27. Lerner J, Tirole J (2002) Some simple economics of open source. J Ind Econ 50(2):197–234
28. Lima A, Rossi L, Musolesi M (2014) Coding together at scale: GitHub as a collaborative social network. In: Proceedings of the international AAAI conference on web and social media, vol 8
29. Ma Y, Bogart C, Amreen S, Zaretzki R, Mockus A (2019) World of code: an infrastructure for mining the universe of open source VCS data. In: 2019 IEEE/ACM 16th international conference on mining software repositories (MSR). IEEE, pp 143–154
30. Matherly J (2015) Complete guide to Shodan. Shodan, LLC (2016-02-25) 1
31. Mergel I (2015) Open collaboration in the public sector: the case of social coding on GitHub. Gov Inf Q 32(4):464–472
32. Murić G, Abeliuk A, Lerman K, Ferrara E (2019) Collaboration drives individual productivity. Proc ACM Hum-Comput Interact 3(CSCW):1–24
33. Murphy SN, Dubey A, Embi PJ, Harris PA, Richter BG, Turisco F, Weber GM, Tcheng JE, Keogh D (2012) Current state of information technologies for the clinical research enterprise across academic medical centers. Clin Transl Sci 5(3):281–284
34. Payne A, Singh V (2010) Open source software use in libraries. Libr Rev 59(9):708–717
35. Pearce JM (2012) Building research equipment with free, open-source hardware. Science 337(6100):1303–1304
36. Perkel J (2016) Democratic databases: science on GitHub. Nat News 538(7623):127
37. Rabai BA et al (2015) Programming language use in us academia and industry. Inform Educ 14(2):143–160
38. Spadini D, Aniche M, Bacchelli A (2018) PyDriller: python framework for mining software repositories. In: Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering - ESEC/FSE 2018. ACM Press, New York, pp 908–911
39. Stevens M, Bursztein E, Karpman P, Albertini A, Markov Y (2017) The first collision for full SHA-1. In: Annual international cryptology conference. Springer, Berlin, pp 570–596
40. Strobl C, Boulesteix AL, Zeileis A, Hothorn T (2007) Bias in random forest variable importance measures: illustrations, sources and a solution. BMC Bioinform 8(1):25
41. Thung F, Bissyande TF, Lo D, Jiang L (2013) Network structure of social coding in GitHub. In: 2013 17th European conference on software maintenance and reengineering. IEEE, pp 323–326
42. Tutko A, Henley A, Mockus A (2020) More effective software repository mining. arXiv preprint. arXiv:2008.03439
43. van Rooij SW (2009) Adopting open-source software applications in us higher education: a cross-disciplinary review of the literature. Rev Educ Res 79(2):682–701
44. Zagalsky A, Feliciano J, Storey MA, Zhao Y, Wang W (2015) The emergence of Github as a collaborative platform for education. In: Proceedings of the 18th ACM conference on computer supported cooperative work & social computing, pp 1906–1917
45. Zöller N, Morgan JH, Schröder T (2020) A topology of groups: what GitHub can tell us about online collaboration. Technol Forecast Soc Change 161:120291