Productivity Spillovers among Knowledge Workers in Agglomerations: Evidence from GitHub*

Lena Abou El-Komboz†, Thomas Fackler‡

May 26, 2023

Abstract

Software engineering is a field with strong geographic concentration, with Silicon Valley as the epitome of a tech cluster. Yet, most studies on the productivity effects of agglomerations measure innovation with patent data, thus capturing only a fraction of the industry's activity. With data from the open source platform GitHub, our study contributes an alternative proxy for productivity, complementing the literature by covering a broad range of software engineering. With user activity data covering the years 2015 to 2021, we relate cluster size to an individual's productivity. Our findings suggest that physical proximity to a large number of other knowledge workers in the same field leads to spillovers, increasing productivity considerably. In further analyses, we confirm the causal relationship with an IV approach and study heterogeneities by cluster size, initial productivity and project characteristics.

Keywords: agglomeration effects, knowledge spillovers, open source, online collaboration
JEL: D62, J24, O33, O36, R32

*We thank Florian Englmaier, Oliver Falck, Moritz Goldbeck, Anna Kerkhof and Chris Stanton for valuable comments and suggestions. We also thank conference participants at EEA2021, EARIE2021, VfS2021, 16th North American Meeting of the Urban Economics Association; and participants at CRC Retreat Schwanenwerder 2021, CESifo/ifo Junior Workshop on Big Data 2021 and an ifo internal seminar.
Support by Deutsche Forschungsgemeinschaft through CRC TRR 190 (project number 280092119) and by the bidt Think Tank project "Changing workplaces: Patterns and determinants of technology and skill adoption by firms and individuals" is gratefully acknowledged. Thomas Fackler thanks the Laboratory for Innovation Science at Harvard for their hospitality while writing parts of this paper.

†ifo Institute and LMU Munich; abou-el-komboz@ifo.de.

‡ifo Institute, LMU Munich, CESifo, and Laboratory for Innovation Science at Harvard, Harvard University; fackler@ifo.de.

1 Introduction

Urban density has a positive impact on wages, worker productivity, and firm productivity. One of the reasons for this relationship is improved diffusion of knowledge through physical proximity (Jaffe et al., 1993; Glaeser, 1999; Atkin et al., 2022). Knowledge spillovers occur among workers, where individuals benefit from the diverse skills of their coworkers and learn from each other, leading to an increase in productivity (Cornelissen et al., 2017). This effect is particularly prominent in innovative sectors, as workers and firms within a research field or industry tend to locate near each other to facilitate collaboration and knowledge exchange (Carlino and Kerr, 2015; Moretti, 2021).

Most previous studies on exposure to innovation and agglomeration effects have relied on patents as a measure of productivity (Carlino et al., 2007; Carlino and Kerr, 2015). However, using patent data provides an incomplete view of innovative activity. Patent filings are only observed with delay after a successful invention. Patents may have little market value and never be put into production, thus resembling an invention but not necessarily an innovation. Lastly, patentability differs across types of ideas and fields, such that the extent to which innovative output is captured varies significantly(Carlino and Kerr, 2015; Cohen and Lemley, 2001). By contrast to the small share of knowledge workers that files patents, coding is a much more widespread activity. Our study contributes to the existing literature by focusing on programmers on GitHub, the world's largest open-source platform, using a novel measure of productivity, the number of code changes called commits, to capture agglomeration effects on productivity. GitHub provides a perfect environment to explore this question in the field of computer science due to the availability of fine-grained data on user interactions and the integrated social features designed to support collaboration (Laurentsyeva, 2019). We complement the literature by analyzing the activity and output of programmers on GitHub, capturing even small shifts in productivity resulting from changes in cluster size.

GitHub offers comprehensive data on the history of interactions among users and information about the users themselves, which is publicly accessible through GitHub Torrent (GHTorrent) (Gousios, 2013). By examining the interactions and social features within the platform, we can investigate the spillover effects on productivity, which are known to be significant in collaborative settings (Catalini, 2018; Azoulay et al., 2010).

In our study, we build upon the empirical approach introduced by Moretti (2021) to examine the agglomeration effects on the productivity of GitHub users in the USA and Canada. We utilize exogenous variations in cluster size resulting from users moving across cities and users joining or leaving a specific technology. This allows us to estimate the impact of changes in technology-specific cluster size on users' productivity in the respective technology, considering both the quantity and quality of their output. Additionally, we explore potential heterogeneity in these effects based on factors such as cluster size and the initial productivity level of individual users.

Estimating agglomeration effects on productivity poses challenges such as simultaneity and unobserved productivity shocks (Combes et al., 2010). To address these concerns, we employ an instrumental variable approach. The geographic network of GitHub projects enables us to predict local cluster size through a shift-share analysis. This approach ensures that the variation in cluster size is independent of local productivity shocks, mitigating any potential biases in estimating the elasticity of productivity and cluster size.

Our findings indicate that cluster size has a positive impact on a user's productivity and the quality of their output. Specifically, a ten percent increase in cluster size within a technology is associated with a 2.8 percent increase in user activity in that technology. Our heterogeneity analysis suggests that the effects are more pronounced for older projects and projects with a smaller share of commits made during business hours.

Open-source software plays a crucial role for firms, leading to an increase in value-added productivity when incorporated into their operations (Nagle, 2019). By studying GitHub users, we can observe the productivity gains resulting from agglomeration effects in a high-tech sector, such as software engineering, where patenting is less pronounced (Cohen and Lemley, 2001).

Cluster effects on productivity, as estimated by Moretti (2021) using patent data, are observable with GitHub data as well. With larger cluster sizes, more users of a given cluster reside in a city and the relocated inventor has a higher chance to encounter and interact with a greater number of users in her cluster and collaborate with them. This might lead to the positive relationship between cluster size and number of commits to projects on GitHub we observe. Clustering among open source software (OSS) contributions seems to be even more profound than for other knowledge workers (Wachs et al., 2022). Our results suggest agglomeration effects and knowledge spillovers as potential drivers of this geographic concentration.

Thus, we contribute to the literature on agglomeration effects on productivity by using commits as a novel proxy for productivity. As research suggests, agglomeration effects are quite profound and even more so for skilled workers and knowledge-intensive tasks (Carlino et al., 2007; Combes et al., 2010; Andersson et al., 2014) and, as our results show, occur for GitHub users as well. Contributing to an existing code base requires a certain level of knowledge, such that GitHub users are often highly skilled. Based on this research design, it is possible to analyze smaller shifts in productivity due to changes in cluster size than in patent data. By contrast to patents, commits are small steps in the innovative process that are observed frequently and with an exact timestamp. Productivity gains caused by agglomeration effects, e.g. after a move to a larger high-tech cluster, can thus be more accurately captured on GitHub projects.

The next section provides a brief overview of the existing literature on innovation and agglomeration economics. Thereafter, the empirical framework is presented in Section 3. The setting and data on GitHub are described in detail, and the estimation strategy to identify agglomeration effects on users'

productivity is explained. The findings of the empirical analysis are presented in the Section 4. Finally, we conclude in Section 5.

2 Related Literature

Our study relates to the literature about peer effects on innovation. This literature studies how peers at different stages in life affect the productivity of the individual. Researchers have investigated how this may differ by setting or task and identified as potential channels knowledge spillovers, social pressure or contagious enthusiasm (Mas and Moretti, 2009; Azoulay et al., 2010). Most relevant for our study is the channel of knowledge spillovers as it was found to matter most for innovative activity (Azoulay et al., 2010). Spillover effects on innovative activity seem to be, besides others, technology class-specific (Bell et al., 2019), motivating the technology-specific cluster definition we use in our research design. Next to that, they occur with a higher chance among colocated knowledge workers and, importantly among colocated collaborators (Catalini, 2018). This further supports our research design focusing on analyzing knowledge spillovers among colocated GH users within a technology.

While these studies provide a good understanding of the importance of knowledge spillovers for innovative activity, the majority uses patent data or wage data for estimation (Carlino and Kerr, 2015). Wage data was found to incompletely capture knowledge spillovers when used as a proxy for productivity (Cornelissen et al., 2017). Patent data as an alternative, represents an invention, however, that does not necessarily entail an innovation (Carlino and Kerr, 2015). The latter, though, increases economic growth (Schumpeter, 1939). Therefore, higher patent activity in a location does not necessarily imply higher monetary gains from new products (Carlino and Kerr, 2015). The majority of open source projects contain real-world applications (Borges et al., 2016), e.g., desktop applications for end users or libraries for other programmers. Therefore, we provide a novel proxy for productivity to measure knowledge spillovers among peers when using GitHub data.

Additionally, we contribute to the field of urban economics. Geographical proximity is the focus of the literature on agglomeration effects. Agglomeration effects describe the benefits of a high urban density on several aspects such as wages, productivity, or incomes (Andersson et al., 2009; Rosenthal and Strange, 2020). Agglomeration economics tries to identify the underlying determinants resulting in the advantages of cities (Giddings, 1890; Duranton and Puga, 2001; Charlot and Duranton, 2004; Combes and Gobillon, 2015).

Agglomeration effects have been analyzed for a long time. Carlino and Kerr (2015) provide a summary of the work in the field. In the seminal paper by Jaffe et al. (1993) the authors use patent citations to measure knowledge spillovers. The researchers find a strong localization of patent citations, which is quite persistent over time. Similarly Carlino et al. (2007) show that living in a denser cluster fosters knowledge production.

Most related to our study is the research by Moretti (2021). Using patent data, he shows that technology-specific cluster size and the productivity of top inventors in the respective technology are positively related at the metropolitan level. This likely applies to a broader set of knowledge workers, such as software engineers, as well. With a move to a larger cluster, surrounded by a larger number of users in her technology, the focal user might experience an increase in productivity caused by knowledge spillovers. By observing all public activity of the users we complement the study by Moretti (2021) with a potentially more precisely measured elasticity between cluster size and productivity in the field of software engineering.

3 Research Design

Now turning to the empirical framework, first, we explain the platform GitHub in more detail. Then, we discuss the steps for data preparation as well as describing the data and end with presenting the estimation strategy.

3.1 Setting: GitHub

GitHub is the world's biggest code hosting site and is based on the `git` revision control system (GIT, 2021). The platform launched in 2008 and since then experienced an increase in popularity among software developers (Fackler et al., 2020). A free basic version and its ease of use made it attractive for users. The platform exhibits features of a social network in line with its motto: "GitHub: social coding" (Lima et al., 2014). After registration, users can create a project to which code can be *pushed*, i.e., uploaded. The platform supports every programming language. Each project has one owner. A *commit* represents the sum of code changes a user sends to the project in a session (Lima et al., 2014). Regarding the social features of GitHub, it is possible to *star* a project. This way, it is bookmarked and can be found more easily later in time. The number of stars per project is seen as a measure of a project's quality and popularity among users (Lima et al., 2014).

When registering, users are able to provide a name, location and other biographical information. Each project can be set private or public. The data used in the analysis contains only commits to public projects. In this case, any actions taking place in a project are observable by everyone (Laurentsyeva, 2019).

The motivation for contributions on OSS platforms spreads from career concerns in the sense of building reputation, paid work at software companies to working on own software projects or helping others (Belenzon and Schankerman, 2015; Hergueux and Jacquemet, 2015)

Social connections play also a role on the platforms. Users are more likely to join the projects of users they have social connections with (Casalnuovo et al., 2015) and when forming new teams, individuals

especially value previous collaborations in self-organized networks. In that case, they gain from past interactions, as they were able to build mutual trust and have a certain level of knowledge about the other's abilities (Casalnuovo et al., 2015). In a larger cluster with a higher chance of encounters, this process may be enhanced.

Thus, agglomeration effects may also affect the productivity of software engineers. Especially in GitHub projects with a small number of project members, users tend to be geographically close to each other (Casalnuovo et al., 2015). Users committing more to projects by other local users may follow a similar pattern as the one identified by Jaffe et al. (1993), with local inventors citing patents by other local inventors more frequently.

3.2 Data

3.2.1 Data Generation

We use a combined version of several snapshots from GitHub Torrent (GHTorrent) (Gousios, 2013). GHTorrent creates snapshots of the public activities on GitHub, e.g. user registration, projects and commits, and makes it accessible in a relational database. The commits recorded are only commits to public repositories. The included approximately biannual snapshots cover the time between September 2015 and March 2021.[1] The activity stream of commits as well as data on the corresponding projects stem from the latest snapshot from March 2021. We limit the commits queried to users that have a US or Canadian location stated in the respective snapshot. [2] The commit data contains all public commits a user has ever generated since account creation until the date of the snapshot.

We assign a project's programming language to any commit committed to that project[3] "Programming language" is understood in a broad sense and includes frameworks and databases. For the analysis, we consider only projects with a stated programming language. The data contains further information about projects, e.g. project owner or number of watchers.

In total, there are 404 stated programming languages in the commit dataset. However, the top 18

[1]More precisely, the snapshots in our data were taken on the 2015/09/25 (201509), 2016/01/08 (201601), 2016/06/01 (201606), 2017/01/19 (201701), 2017/06/01 (201706), 2018/01/01 (201801), 2018/11/01 (201811), 2019/06/01 (201906), 2020/07/01 (202007) and 2021/03/06 (202103).

[2]Every user has a unique user id. Commits are matched via the author id, not the committer id, to the user id. The objective of the analysis is to examine productivity changes based on variations in cluster size, specifically focusing on whether users generate more (new) output as indicated by an increase in commits. It is more likely that increased productivity will be reflected in written commits rather than uploaded commits. Matching commits based on the committer id might capture a higher level of activity on GitHub in general, but the connection to higher productivity is less evident. It is possible that a user creates numerous pull requests with content authored by other users. Consequently, matching commits based on the author is provides a better measure of the user's knowledge output. It should be noted that some users possess multiple GitHub accounts (Casalnuovo et al., 2015). Unfortunately, we are unable to account for these user aliases, which could result in an underestimation of spillover effects due to fewer commits being attributed to a user than she actually contributes. Nonetheless, relative increases in productivity should remain unaffected on average.

[3]A project can have files in several programming languages. GHTorrent defines the project's programming language as the programming language that makes up the largest number of bytes in the project.

programming languages cover 90 percent of all commits.[4] Defining clusters based on all programming languages may result in a sizable number of clusters with only a single user of a particular programming language. Moreover, different programming languages can be closely related, and users can derive benefits across different programming languages from the knowledge of others. To better capture these spillover effects, we account for such correlations by employing a grouped technology definition. Hence, we assign programming languages to specific technology classes based on the Stack Overflow Developer Survey 2020 (Stack Overflow, 2020). In the survey clusters of databases, programming languages, frameworks and platforms were created. They show which technologies are frequently used together by developers.[5] We focus on the top 18 programming languages and match them to these groups, which, in the end, leads to five technology fields. Technology one contains JavaScript, CSS, HTML, PHP, C# and TypeScript. Technology two Python, Shell, Go, Jupyter Notebook and R, technology three Ruby, technology four Java, Objective-C and Swift, and technology five C++, C and Rust.[6]

Commits are aggregated to snapshot intervals. The first time interval comprises all commits to a project after the user account was created and up to 25 September 2015. The second interval contains all commits to a project between 26 September 2015 and 8 January 2016. This system follows for the other snapshot intervals and results in ten time intervals.[7]

To remove inactive accounts, we restrict the data to users that commit in at least two time intervals. If the account was created in the last time interval and the user committed in that time interval, those users are included as well.[8]

Each snapshot contains only the currently stated location. We combine the snapshots of the user accounts from 201509 until 202103 to observe user location changes.[9]

Users' location in our data is self-stated. In about 90 percent of cases, i.e. 210,705,552 user-snapshot observations, this variable is missing as users did not state any location. For these cases, we try to fill in the location data from other snapshots (if possible, first from the previous snapshot, and otherwise from the next one). Based on the stated location we then geocode the user.[10]

We further match users to one of the 179 US "Economic Areas" defined by the Bureau of Economic Analysis (BEA) or the Canadian equivalent, namely one of the 76 economic regions defined by Statistics

[4]These are C, C#, C++, CSS, Go, HTML, Java, JavaScript, Jupyter Notebook, Objective-C, PHP, Python, R, Ruby, Rust, Shell, Swift, and TypeScript.

[5]A visualization of the correlated technology clusters can be found here (last accessed 17 March 2023): https://insights.stackoverflow.com/survey/2020#correlated-technologies

[6]Technology three consists only of Ruby, however the programming language is one of the five most used programming languages. Thus, not including Ruby would lead to excluding a large amount of commits.

[7]In the following, we will use the terms snapshots and time intervals as equivalents.

[8]There are two types of accounts, users and organizations. Organizations, a group of users that appear as meta users, can only own projects, but cannot do any other actions. Therefore organization accounts are excluded.

[9]To the best of our knowledge, our study is the first to construct a panel of GitHub users to analyze mobility events.

[10]The *location* variable is matched with data sets on "us.cities", "canada.cities" and "world.cities" provided by the R-package *maps* (Becker and Wilks, 2018), which contain coordinates of cities. Furthermore, for US and Canadian cities, more comprehensive data sets provided for free by simplemaps was used (Simplemaps, 2021).

Canada. In many cases, "Economic Areas" are comparable to Metropolitan Statistical Areas (MSA). However in the case of larger areas such as the San Francisco Bay Area or New York, the "Economic Area" covers the entire economic region and, thus, is larger than the corresponding MSA. In the following, Economic Areas are called "cities". Finally, the user data contains 1,017,332 users with (always) US or Canadian locations, matched to economic areas, with 7,448,824 user-snapshot observations.

3.2.2 Final data set

The combined commits and user data results in 12,215,907 user-project-snapshot observations. We calculate cluster size based on the full data set. For the regression, we use only users that are observed in all snapshots, i.e. geocoded and with non-zero commits in all time intervals, such that 2,527,496 observations and 21,116 unique users remain.

In Section 3.3 we describe the regression data in detail. It is very similar regarding the distribution of commits per programming language and per technology compared to the full data. On the other hand, the projects, users commit to, tend to have more stars and users included are more active regarding their number of commits in the regression data. Thus we are more likely to observe productivity effects for those users in larger clusters. For users that generally commit less, it is harder to identify an increase in their commits on GitHub. They might experience positive productivity spillover effects from denser local clusters, though this might not result in a higher number of commits, as they were less active on GitHub to start with.

Calculating cluster size using the regression data might measure cluster size less precisely. If a user in our data does not commit in a time interval, it might be the case that she commits to a private project. Even if the user does not commit at all in a time interval, she might still have positive productivity spillover effects on the other active users.

For robustness, we estimate the elasticity between cluster size and productivity loosening the restriction on the length of time intervals with non-zero commits per user. In this specification, the elasticity becomes less significant.[11]

3.2.3 Clusters

The clusters are constructed as technology $\times$ city. Cluster size S for user i in time t in technology f in city c is the number of users in technology f in city c excluding user i relative to all users in a technology f in time t. More formally, cluster size is calculated as:

$$S_{-ifct} = \frac{\sum_{j \neq i} N_{jfct}}{\sum N_{jft}}$$

[11] See Section 4.6 for the discussion of the results.

where the summation of users N is across all users j in city c in technology f in time t but user i. Cluster size is defined in relative terms by dividing the sum of users in city c in technology f excluding user i by the total number of users N in technology f in time t.

The technology of a user in a snapshot is determined by the projects a user commits to. In practice, a user that commits to projects in the technologies 1 and 2 in the first time interval, is assigned to the clusters 1 $\times$ city and 2 $\times$ city in that interval.

The accuracy of cluster size calculation relies on users providing correct location information and maintaining up-to-date profiles. Thus it depends on how regularly users update their accounts. Less active users may update their information less frequently. However, these users may derive fewer benefits from denser clusters, or at least it may be more challenging to identify such benefits because a smaller number of commits is observed for them and location changes may be delayed. Conversely, if these users become more productive and commit more after moving without updating their location, it introduces a downward bias in our estimates. In such cases, the relevant cluster size would be inaccurately measured, as the user would be assigned to a city where they are no longer residing. Consequently, this discrepancy potentially introduces measurement error into our analysis.

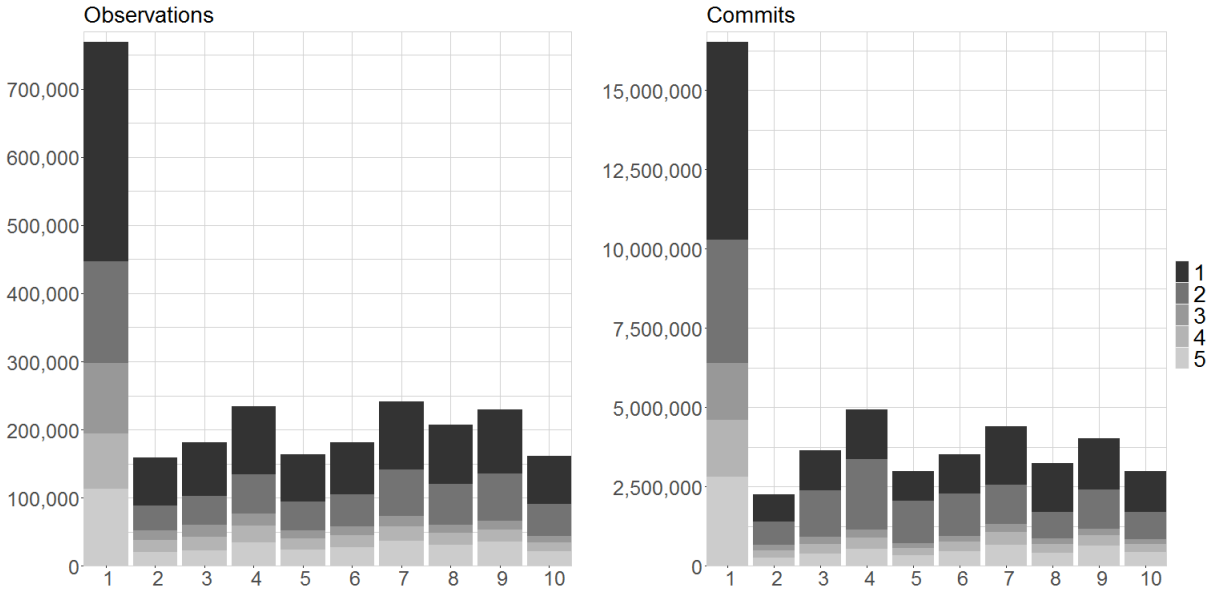
3.3 Descriptive Analysis

In this section, we provide a descriptive analysis of the regression data, in which only users with commits in all time intervals are included.

3.3.1 Commits, Users and Projects

The left graph of Figure 1 shows the number of observations per time interval and per technology. Our level of observation is the number of commits to a project by a user in a time interval. Part of the variation in the number of observations is due to the different lengths of the time intervals. For example, the first, seventh and ninth time interval capture a longer time period in comparison to the other time intervals. The peak in observations in the first time interval is due to the fact, that it contains all commits to projects after account creation until 25 September 2015. This time interval captures the longest period of time and results in the greatest number of observations per time interval. The seventh and ninth time intervals are about a year long, whereas the other time intervals are about six month long and, thus, represent a larger number of commits. For the regressions, we include time fixed effects to take this variation of commits into account.

Figure 1: Number of Observation or Commits per Snapshot and per Technology



Note: The left plot shows the number of observations per technology and time interval. The right plot shows the sum of commits per technology and time interval. Sources: GHTorrent, own calculations.

Technologies one and two contain a higher number of observation and sum of commits as they are comprised of a larger number of programming languages, shown in Figure 1.[12] Remarkably, technology three, containing only Ruby, is very similar in distribution to technology four, which contains three programming languages.[13]

This is also noticeable by median number of total commits per technology shown in Table 15 in the Appendix. Technology one makes up about one third of all commits (37.85%), followed by technology two (31.33%), three (7.27%), four (9.25%) and five (14.29%). Hence, even though technologies vary in their number of programming languages, their distribution in the data is relatively similar. Only technology one makes up a very large share. However, it also contains JavaScript, the most used programming language, which by itself already makes up about 18.23% of all commits.

Commits can be split into those to a user's own projects and those to others' projects. In our data, 56% of commits are contributed to projects of other users, while the remaining 44% go to users' own projects.[14] This suggests that the projects in our sample are not necessarily 'toy' projects or projects a user might only use for saving files.[15] The project age, i.e. the time in years since it was created until

[12]Cluster one contains JavaScript, CSS, HTML, PHP, C# and TypeScript. Python, Shell, Go, Jupyter Notebook and R are in cluster two.

[13]Namely Java, Objective-C and Swift.

[14]Project ownership is determined if author id is equal to owner id. Hence, there is a potential of misclassification. A programmer might have several accounts and, thus, several user ids. A project may be owned by one id and commits are done by another id. Unfortunately, we cannot check for that aspect.

[15]To clean for 'toy' projects, Prana et al. (2021) restrict their sample to projects which existed for at least 180 days. In our analysis, this restriction does not make a difference to the estimates.

the latest snapshot date, supports this assumption. In our sample projects are about five years old. As OSS projects go through different stages in their development process, an older project is likely more mature.[16]

Firms often manage OSS projects (Riehle, 2012). The mean share of commits made during business hours, i.e. Monday through Friday from 6am to 20pm, across projects supports this hypothesis. About two third, i.e. 62% of commits, are made during business hours. On the other hand, the share of commits made at the weekend is only 19%. So, it seems the projects tend to mainly be work projects. For leisure projects, cluster size may matter more because users are not in a set team where knowledge exchange is organized by the firm. Therefore, we let the elasticity between cluster size and productivity vary by the share of commits made during business hours.

The number of stars per project are seen as a measure for the quality of a project (Laurentsyeva, 2019). More than half of the projects in our sample do not have any stars. The distribution of stars is highly skewed and the project with the most stars has 259,118. This suggests large variation in the quality of the projects. Focusing on commits to projects with a large number of stars may mitigate the concern of low quality commits. These projects likely require a higher quality of contributions. We thus focus in an additional analysis on commits to projects with a large number of stars to estimate an increase in commit quality with an increase in cluster size. If we also observe a larger elasticity between user activity and cluster size for those projects, commit quality may also increase with cluster size.

Table 16 in the Appendix shows the total number of commits per user, considering all programming languages. [17] The average total number of commits is 2,298.32 compared to the median of 1,059 total commits per user. This large difference between median and mean suggests strong variation in the users' activities. We look into heterogeneity with respect to user activity level. Productivity gains with an increase in cluster size may be better captured for the most active users as we observe more (or all) of their activity on the platform.

Out of the 21,116 users, 4,598 users moved in sum 5,527 times over the whole observation period.[18] This suggests sufficient variation in cluster size.

On average, users are active in more than one programming languages. The average number of programming languages used in total is 7.03 (median: 7). Per time interval, the average number of

[16] E.g. SourceForge, another popular OSS platform has development status categories. These are *production/stable*, *beta*, *planning*, *alpha*, *pre-alpha*, *inactive*, *mature*. Unfortunately, SourceForge does not give a definition of the status with which we could classify our projects.

[17] In some research (Casalnuovo et al., 2015) large commits are excluded as they might not capture typical developer behavior. As Hindle et al. (2008) show, both small and large commits are important steps in a project's development and capture productive output. They find, that small commits are more of corrective nature, for example bug fixes. Large commits tend to be of perfective nature, such as code clean-up or changing the format of the code. Therefore, large commits tend to affect the whole architecture of the project code.

[18] 3,763 users moved once, 752 users moved twice, 72 users moved three times and 11 users moved four times. Moves occurred in the second time interval 1,484 times, 263 times in the third time interval, 773 times in the fourth time interval, 1,466 times in the seventh time interval and 1,541 times in the tenth time interval.

programming languages is 3.24 (median: 3). Regarding technologies, users are on average active in 3.42 (median: 3) technologies and per snapshot in 2.25 (median: 2) technologies. Calculating cluster size by technology thus may better capture the field a user is mainly active in compared to using programming language as cluster definition. Next to that, it also indicates that users start and stop using a technology during our observation period, which adds a second source of variation in cluster size, in addition to users moving.

3.3.2 Clusters

Now we turn to describing the clusters as well as changes in cluster size over time. Clusters are calculated on the basis of all users, i.e. also users not being observed with commits over the whole observation period.

In the Appendix, Table 17 shows the largest clusters for the five technologies as of March 2021 (202103, the latest snapshot) and Table 18 shows the distribution of cluster size per technology.

For all technologies, San Jose-San Francisco-Oakland and New York-Newark-Bridgeport are the two largest clusters. Especially for San Jose-San Francisco-Oakland, between 10 to 14 percent of all users in the respective technology in the snapshot 202103 are located in this area.

In the case of technology one, San Jose-San Francisco-Oakland makes up about 10.64 percent of all users in that technology, followed with a considerable gap by New York-Newark-Bridgeport with 8.78 percent and next Seattle-Tacoma-Olympia with 5.37 percent. The top ten cities cover 49.62 percent of all users in technology one in the latest time interval. The ratio between the largest cluster and the median cluster is 236.4. This means about 236 times more users stem from San Jose-San Francisco-Oakland than from the median cluster. The ratio between the 90th percentile and the median cluster size for technology one is 23.42. This shows that moving away from the largest cluster, cluster size decreases rapidly.

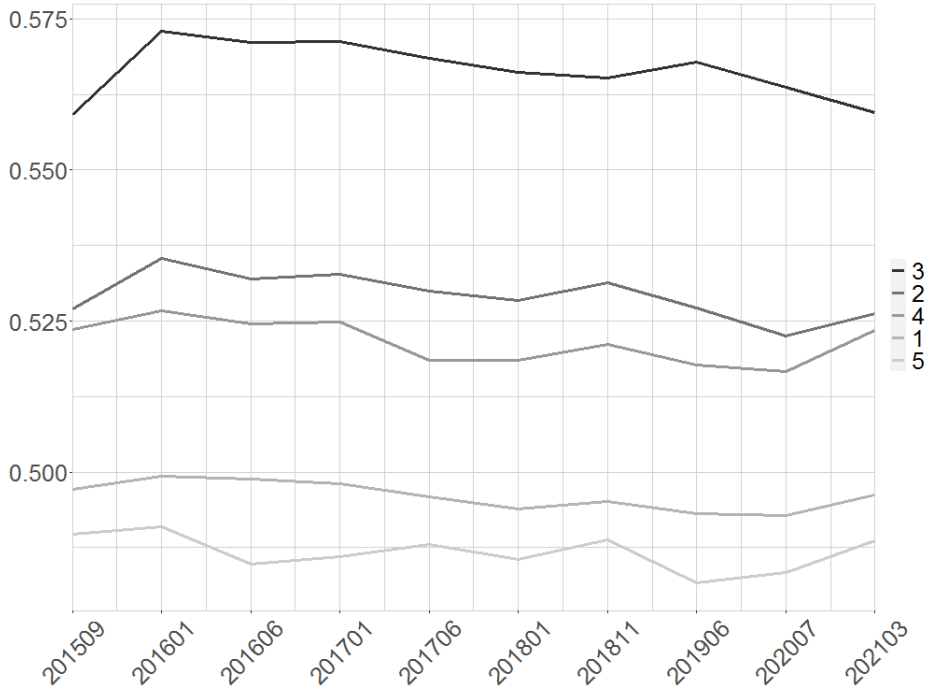
For technology two, San Jose-San Francisco-Oakland is again by far the largest cluster with 13.44 percent of all users. The ratio between largest cluster and the median cluster is 280, i.e. 280 times more users stem from San Jose-San Francisco-Oakland than from the median cluster, with 0.04 percent. At 21.71 the ratio between the 90th percentile and the median cluster is much smaller. The difference between largest cluster and 90th percentile cluster is even more extreme for technology two than for technology one.

In the case of technology three, four and five again San Jose-San Francisco-Oakland is the largest cluster with 14.15 percent, 13.4 percent, and 13 percent respectively. Ratios between largest and median cluster are 257.35, 279.27 and 217.5.

As noticeable in Figure 2, users in the technologies are already quite concentrated to start with. The figure plots the share of users originating from the top ten cities relative to all users in the respective technology. Especially for technology three with about 55 percent of all users in that technology stemming from only ten cities and technology two with about 53 percent in the tenth time interval, clustering seems

to be profound. However, for most technologies the share over time does not change much.

Figure 2: Share of Top 10 Cities for All Technologies



Note: Plot shows the share of users stemming from the top ten cities relative to all users per technologies for all technologies over time. Sources: GHTorrent, own calculations.

3.4 Estimation Strategy

To study the relationship between cluster size and productivity, we implement the following regression equation:

$$ln(y_{ijflct}) = \alpha\, ln(S_{-ifct}) + d_{cf} + d_{cl} + d_{lt} + d_{ct} + d_i + d_j + \mu_{ijflct} \tag{1}$$

where y_{ijflct} is the number of commits of user i in time interval t to project j located in city c in the technology f and programming language l; S_{-ifct} is the cluster size in city c of the technology f in time interval t, excluding user i; d_{cf} are city $\times$ technology effects, controlling for city specific effects in technologies; d_{cl} are city $\times$ programming language effects, controlling for city specific effects in programming languages; d_{lt} are programming language $\times$ time effects, accounting for trends in programming languages; d_{ct} are city $\times$ time effects, taking into account changes in cities over time; d_i controls for time-invariant individual effects and d_j for time-invariant project effects. Standard errors are clustered on the city $\times$ technology level to take into account serial correlation. Variation in cluster size stems from users moving and by users starting and stopping to commit in a technology.

If the coefficient α is positive, it indicates that there is a positive relationship between cluster size and productivity. This means that as the size of the cluster increases, either through more local users using the technology or by moving to a larger cluster, the user's productivity also increases. This suggests that there are spillover effects that contribute to the user's increased productivity, as more users are positively associated with the focal user's activity. By contrast, a negative α would suggest that the user commits less as cluster size increases.[19]

Clusters are defined as the number of users in a city relative to all users in a technology. By including city $\times$ time fixed effects, we take into account changes in city size when estimating the effect of cluster density on productivity.

3.5 Instrumental Variable Approach

An endogeneity concern when estimating agglomeration effects are unobserved determinants in the error term μ_{ijflct} simultaneously affecting productivity and cluster size (Combes and Gobillon, 2015). Time-invariant characteristics of a city biasing the estimates of α, e.g. location of the city, are controlled for by city effects. The attractiveness of a city or its size, which may change over time, is controlled for by city $\times$ time effects. Trends in the popularity of programming languages or technologies are taken into account by programming language, technology and programming language $\times$ time fixed effects. Moreover, differences in commits due to the popularity of projects are not affecting the estimates by including project fixed effects.

A possible concern in our case could be that users move to a larger cluster expecting to be more active there. The user fixed effects control for users' inherent level of activity, although not for changes in their ability. In that case, unobservable time-varying productivity shocks could both affect the cluster size and the user's number of commits. Either via sorting, i.e., endogenous quality of labor, or simultaneity (endogenous quantity of labor), the OLS results might be biased.

Another concern regarding OLS identification of α are unobserved productivity shocks at the individual or local level. In a city, firms may start to cluster all activity in one technology. Users then might start to commit in that technology, both cluster size and productivity would increase, however caused by unobserved productivity shocks at the city $\times$ technology level.

Using an instrumental variable approach similar to Moretti (2021), we instrument the changes in local cluster size by changes that originate elsewhere to remove any possible bias due to unobservable productivity shocks affecting both outcome variable and the coefficient of interest.

A valid instrument should be first relevant, meaning in this case, it should be a good predictor for changes in local cluster size. Second, it should be exogenous to unobserved local cluster characteristics

[19] A limitation of our data is that we can only observe commits to public projects, but not changes in contributions to private projects on GitHub. Thus, part of the change in observed activity on public projects could be caused by shifts from/towards private projects (or other platforms than GitHub).

and not an outcome of the dependent variable, i.e., the productivity of a user in a local cluster (Combes and Gobillon, 2015). Changes in the activity of local GitHub projects originating elsewhere arguably meet these conditions. If a project attracts more users committing from outside the local city, it is likely that the number of local users committing to the project will also increase. These increases in user activity elsewhere can serve as good indicators of changes in the number of local users, and subsequently, the local cluster size. At the same time, these expansions elsewhere are unlikely to be an outcome of productivity gains of local GitHub users and any concern of reverse causality is mitigated. Besides, changes in cluster size somewhere else are possibly uncorrelated to unobserved local cluster characteristics. As a result, we obtain an exogenous and relevant instrument for changes in local cluster size.

GitHub is a particularly suitable setting to exploit exogenous sources for an instrumental variable approach. The platform is online, hence users from all across the country can commit to a project. Therefore, sufficient variation in the number of committers from different cities should be available to apply a variation of a shift-share instrument.

In practice, we consider changes in the projects of a technology that other local users are committing to, i.e. not the projects of the focal user, originating elsewhere to predict the local cluster size of the technology for a user. For that, we calculate the sum of changes in users of projects, excluding the local users, and divide it by the total change in users in a technology. Let $N_{jf(-c)t}$ be the sum of users committing to project j in time interval t and technology f excluding city c. Then the change between t and $t-1$ is $\Delta N_{jf(-c)t} = N_{jf(-c)t}$ - $N_{jf(-c)(t-1)}$.

Formally the instrument for the cluster size of user i in cluster fct is calculated as:

$$IV_{ifct} = \sum_{s \neq j_i} D_{sfc(t-1)} \frac{\Delta N_{sf(-c)t}}{\Delta N_{ft}} \tag{2}$$

where $D_{sfc(t-1)}$ is an indicator if project s in technology f was present in c in time interval $t-1$, $N_{sf(-c)t}$ is the log sum of users committing to project s in technology f, time interval t in all cities but city c to which user i does not commit to, then $\Delta N_{sf(-c)t}$ is the change in log users committing to project s in technology f and time interval t for all cities but city c; N_{ft} is the log total sum of users in time interval t in technology f and ΔN_{ft} is the overall change in log users in technology f in time interval t. Summation is across all projects present in city c in technology f but user i's projects j.[20] Identification stems from changes in the number of users originating from other cities committing to local projects besides the focal user's projects.

For example, users from San Francisco and New York commit to a project x. If more users in New

[20] In the shift-share analysis, an aggregate change in a sector is used to predict the local change in a sector. Here, the local change in a technology, i.e., cluster city $\times$ technology, is predicted by the relative changes in users for projects in a technology elsewhere. Additionally, it relies on the presence of the projects in the local city. Then, the instrument is the change in users for projects in a technology elsewhere, but present in the local city, relative to the overall change (in the US and Canada) in users in a technology.

York start to commit to project x, the number of users in San Francisco committing to project x might also increase. This possibly leads to an increase in the number of users committing to other projects in San Francisco in the same technology. As a result, the cluster size would increase in San Francisco. In that case, growth of local cluster size would be driven by changes elsewhere. The increase in the number of users committing to project x in New York is unlikely affected by unobserved productivity shocks of users in San Francisco not committing to project x.[21]

In detail, for user i, variation in the number of users committing to project x in i's technology in other cities than i's city and that i is not committing to herself, is independent of unobserved factors that affect i's productivity conditional on covariates. Programming language $\times$ time and programming language fixed effects take into account general variation in the popularity of programming languages over time and, hence, changes in projects due to that. City fixed effects control for characteristics in a city possibly affecting productivity. User and time fixed effects control for general activity of users and snapshot-specific characteristics. Therefore, identification relies on the existence of projects, other than the user's projects, and changes in users committing to these projects in other cities than the user's city.

The instrumental variable approach predicts changes in cluster size and not its level. To compare its results with the baseline model, the latter has to be estimated using the first differences of equation (1) (with additional fixed effects):

$$\Delta ln(y_{ifct}) = \alpha \Delta ln(S_{-ifct}) + d_t + d_c + d_i + d_l + d_{lt} + \mu_{ijflct} \tag{3}$$

This way, the contemporaneous effect of cluster size on productivity is estimated. It reflects the direct change in the productivity in a technology within a project with a change in cluster size in a technology. The estimate may be smaller as changes in productivity with changes in cluster size possibly take more time.

4 Results

This section presents the baseline estimates of equation (1). To address possible endogeneity concerns we implement an IV approach. Additionally, we show the results of a heterogeneity and mechanism analysis as well as robustness checks to test the validity of the results.

[21] More precisely we create a panel of users and projects. A user is also considered to be connected to a project if she committed in a past time interval to a project, not necessarily in the current time interval. By committing to the project in the past, she may still be aware of the activity of the project and the exclusion restriction possibly does not hold.

4.1 Baseline Estimates

Table 1 shows the estimates for the OLS regression of equation (1). For this regression, only users who commit in all time intervals are included.

The estimated elasticity in the first column, conditioning on city, time, technology, programming language, project and user fixed effects, is 0.1144 (0.1099). Trends in programming languages and technologies or productivity shocks for certain programming languages and technologies are captured by programming language $\times$ time fixed effects, decreasing the coefficient for log size from 0.1144 (0.1099) to 0.0928 (0.0744) in column two of Table 1. The decrease in the elasticity after adding controls for time trends in programming languages hints at larger clusters experiencing more positive productivity shocks as a result of the general popularity of the languages most frequently used there.

Table 1: Baseline Estimates

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1144	0.0928	0.1966**	0.1934**	0.2775**
	(0.1099)	(0.0744)	(0.0949)	(0.0962)	(0.1253)
Fixed-effects					
City	Yes	Yes	Yes	Yes	Yes
Time	Yes	Yes	Yes	Yes	Yes
Language	Yes	Yes	Yes	Yes	Yes
Technology	Yes	Yes	Yes	Yes	Yes
Project	Yes	Yes	Yes	Yes	Yes
User	Yes	Yes	Yes	Yes	Yes
Language x Time		Yes	Yes	Yes	Yes
City x Technology			Yes	Yes	Yes
City x Language				Yes	Yes
City x Time					Yes
Adjusted R^2	0.287	0.290	0.291	0.291	0.292
Observations	2,527,496	2,527,496	2,527,496	2,527,496	2,527,496

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression.

After adding city $\times$ technology fixed effects in column three, taking into account time-invariant city-technology characteristics, the coefficient for log size becomes 0.1966 (0.0949) and is now statistically significant at the five percent level. The coefficient in column three becomes more than twice as high as in column two. This suggests that larger clusters are associated with less active users. One possible explanation may be that users in those cities tend to commit more to private projects than public projects, and thus lower activity is observed.[22]

[22]This is also supported by excluding San Jose-San Francisco-Oakland, the largest city for all technologies in the last

The estimates stay significant at the five percent level after adding controls for the interaction of city $\times$ language effects in column four. The elasticity remains almost the same in column four.

When including all controls the estimated elasticity is 0.2775 (0.1253) and significant at the five percent level. This, on the other hand, suggests that city-specific productivity shocks and selection due to local amenities especially seem to matter for smaller clusters.

The final estimate implies a positive elasticity, meaning a user commits 2.8 percent more in a time interval in a technology with a ten percent increase in cluster size of the respective technology. Precisely, if the share of users in the user's city in her technology relative to all users in her technology increases, she commits more in that technology. For example, a user's number of commits in technology one would increase by 19 percent, if she moved from Chicago to Seattle. Hence, there is a positive relationship between cluster size and productivity. This is in line with the findings of others (Moretti, 2021; Combes et al., 2010), that similarly estimate a positive elasticity between cluster size and productivity.

We only observe commits to public projects. If a user's cluster increases, she might move to committing more to private projects. In that case, the results would provide a lower bound of the elasticity. [23]

4.2 Quality of Commits: Project Stars

To analyze if the quality of a user's commits increases with cluster size, we restrict the sample to users that are observable over the whole period of analysis and consider only commits to the top ten percent of projects by number of stars.[24] As the number of stars are a measure for the quality of a project, committing more to those high quality projects suggests an increase in the commit's quality itself. Table 2 shows the results of a regression of log commit on log size with the restricted sample.

The coefficients in the first and second column are insignificant with 0.1451 (0.1043) and 0.1229 (0.0827). In the next columns it becomes significant at the one and five percent level, varying from 0.2649 (0.0859) to 0.3239 (0.1462). In the final column, conditional on all fixed effects, the elasticity between commits and cluster size is significant at the five percent level with 0.3239 (0.1462).

The positive elasticity between cluster size and number of commits of 0.3239 (0.1462) implies that a user commits 3.2 percent more in a technology to projects with at least five stars if her cluster in that technology increases by ten percent. This suggests a positive impact of cluster size on the quality of commits.

snapshot, from the sample. The final elasticity becomes even larger with 0.3219 (0.1283) in comparison to 0.2775 (0.1253), both significant at the five percent level.

[23] Additionally, we test if the results are stable for excluding large numbers of commits and large projects, i.e. more than 100 commits and projects with more than 40 users committing to. The elasticity slightly increases to 0.2771 (0.1168) and remains significant at the five percent level. See Table 19 in the Appendix for the regression results.

[24] Projects in the upper ten percent of the stars per project distribution have at least five stars.

Table 2: Baseline Estimates - Upper 10% of Projects

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1451	0.1229	0.2649***	0.2637***	0.3239**
	(0.1043)	(0.0827)	(0.0859)	(0.0867)	(0.1462)
Fixed-effects					
City	Yes	Yes	Yes	Yes	Yes
Time	Yes	Yes	Yes	Yes	Yes
Language	Yes	Yes	Yes	Yes	Yes
Technology	Yes	Yes	Yes	Yes	Yes
Project	Yes	Yes	Yes	Yes	Yes
User	Yes	Yes	Yes	Yes	Yes
Language x Time		Yes	Yes	Yes	Yes
City x Technology			Yes	Yes	Yes
City x Language				Yes	Yes
City x Time					Yes
Adjusted R^2	0.407	0.409	0.410	0.412	0.413
Observations	392,984	392,984	392,984	392,984	392,984

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression. Sample includes only projects in the upper ten percent distribution of stars per project. These are projects with at least five stars.

4.3 Heterogeneity

Cluster Size The results for the elasticity between number of commits and cluster size might differ depending on the cluster size. It could be the case that productivity spillovers require a certain cluster size to occur. In smaller clusters, the benefits of the existence of other users, as they are fewer, might also be smaller. In this case, it may depend on a certain threshold for agglomeration effects to occur. By contrast, in larger clusters, a one percent increase in cluster size might result in smaller productivity gains in relative terms, compared to a one percent increase in a smaller cluster.[25] Finally, both could be true, implying an S-shaped elasticity between cluster size and productivity. Therefore, we let the effect of cluster size on commits vary with respect to cluster size.

Table 3: Heterogeneity in Elasticity by Cluster Size

	Log(Commit)
	(1)
First Quartile (Smallest)	0.2748**
	(0.1250)
Second Quartile	0.2688**
	(0.1258)
Third Quartile	0.2609**
	(0.1272)
Fourth Quartile (Largest)	0.2651**
	(0.1268)
Adjusted R^2	0.292
Observations	2,527,496
Wald (joint nullity), p-value	0.170

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. In all regressions, fixed effects for city, time, programming language, city × programming language, programming language × time, city × technology, technology, city × time, project and user are included.

Table 3 shows the results of a regression of log commits on log size, where the size is interacted with dummies for cluster size quartiles. All controls from the baseline estimation are included.

The coefficient is the greatest for the smallest clusters, i.e., the elasticity is the highest with 0.2748 (0.1250) in the smallest clusters, conditional on all covariates.[26] The estimates for different size quartiles range from 0.2748 (smallest size quartile; 0.1250) to 0.2651 (largest quartile; 0.1268), hence the variation in elasticities across quartiles is rather small.

They suggest a slightly S-shaped elasticity between cluster size and productivity, as the estimates vary with size quartile. A Wald test for testing that all coefficients are zero in the model with all controls

[25] Au and Henderson (2006), e.g., estimate a bell-shaped relation between productivity and city size for Chinese cities.

[26] This also suggests, that especially large clusters as the San Francisco - San Jose area are not driving our results.

included, cannot be rejected with a p-value of 0.170. Hence, the elasticity between cluster size and productivity does not seem to vary with respect to cluster size. This is in line with the findings of Moretti (2021), which also do not find a heterogeneity in elasticity by cluster size.

Project Age The elasticity may vary by the projects' age. OSS projects evolve over time. In the beginning, where work routines evolve, users may benefit more from being surround by more users in a technology they are working in. It may help them to set up the project. On the other hand, in later, phases where the project is more established and, thus, development steps might be smaller, the gains of a larger cluster size may help the user more to further improve the project. Ayoubi et al. (2017) also show that the probability of learning from team members is larger for more established collaborations. The same can possibly be applied to more established OSS projects and learning within clusters. The older, and likely more established, projects enjoy a larger productivity increase with an increase in cluster size.

Table 4: Heterogeneity in Elasticity by Project Age

	Log(Commit)
	(1)
First Quartile (Youngest)	0.2639**
	(0.1228)
Second Quartile	0.2661**
	(0.1248)
Third Quartile	0.2725**
	(0.1268)
Fourth Quartile (Oldest)	0.2899**
	(0.1263)
Adjusted R^2	0.292
Observations	2,527,496
Wald (joint nullity), p-value	0.046

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Project Age is measured by the years after project start until the date of the latest snapshot, 2021-03-06. The oldest projects are projects that are in the fourth quartile of the distribution of project years. In all regressions, fixed effects for city, time, programming language, city $\times$ programming language, programming language $\times$ time, city $\times$ technology, technology, city $\times$ time, project and user are included.

Therefore, we let the elasticity vary by project age. The age of project in years is calculated by the difference between project creation date and the date of the latest snapshot, 2021-03-06. Table 4 shows the results of a regression of log commit on log size letting the estimate vary by project age quartile including all controls from the baseline model. The elasticity increases linearly with project age quartile from 0.2639 (youngest quartile; 0.1228) to 0.2899 (oldest quartile; 0.1263), all significant at the five percent level. A Wald test for testing that all coefficients are zero can be rejected with a p-value of 0.046.

Thus, the elasticity between cluster size and productivity varies with project age, suggesting that the largest knowledge spillovers occur for older projects.

Possibly, for starting a software project more basic knowledge is necessary. With time and development more and more specific knowledge might be necessary to further improve the project. In larger clusters likely more knowledge is available of which especially more advanced projects benefit.[27]

Business Projects A last characteristic of projects which may lead to variation in cluster size is, if it is a leisure or work project. The latter might be established projects led and funded by a firm, potentially requiring contributors to be more knowledgeable about the project and the professional environment. On the other hand, leisure projects are projects, a user likely works on without a fixed team or a firm setting up a plan of tasks that need to be done. Thus, the user (and the project) benefits more from a larger cluster size as users can exchange knowledge without any restrictions and implement new ideas without prior approval from managers.

Table 5: Heterogeneity in Elasticity by Share of Commits made during Business Hours

	Log(Commit)
	(1)
First Quartile (Leisure)	0.2801**
	(0.1238)
Second Quartile	0.2806**
	(0.1261)
Third Quartile	0.2836**
	(0.1254)
Fourth Quartile (Business)	0.2456*
	(0.1254)
Adjusted R^2	0.292
Observations	2,527,496
Wald (joint nullity), p-value	0.006

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Business commits are commits created during work days (monday through friday) and work hours (6am untill 8pm). The projects with the largest share of business commits received are projects that are in the fourth quartile of the distribution of business commits per project. In all regressions, fixed effects for city, time, programming language, city $\times$ programming language, programming language $\times$ time, city $\times$ technology, technology, city $\times$ time, project and user are included.

In Table 5 we let the elasticity between cluster size and productivity vary by the share of commits

[27]The project age may be seen as a proxy for the stage of the *Software Development Life Cycle*. It is generally comprised of six phases: requirements specification and analysis, design, coding, testing, deployment and maintenance. In the earlier phases it is more about setting up the project, whereas in later stages with coding, testing, deployment and maintenance the project is implemented and improved (Carreteiro et al., 2016). This likely is associated with higher commit activity, and thus, productivity.

made during business hours per project. Business commits are commits created during business hours, i.e. Monday through Friday from 6am to 8pm (McDermott and Hansen, 2021). The projects in the fourth quartile have the highest share of business commits, while those in the first quartile have the lowest (and are thus most likely to be leisure projects).

The elasticity, conditional on all controls from the baseline model, is very similar for the first three quartiles with 0.2801 (first quartile; 0.1238), 0.2806 (second quartile; 0.1261) and 0.2836 (third quartile; 0.1254) and significant at the five percent level. For the fourth quartile, the projects with a very large share to only receiving business commits, the elasticity is smaller with 0.2456 (0.1254) and significant at the ten percent level. Even though the difference in estimates are rather small, a Wald test can be rejected with a p-value of 0.006. Thus, elasticity varies with the share of business commits per project.

The larger elasticity for projects with smaller shares of business commits suggests that knowledge spillovers may have a larger effect on leisure projects. One reason for this difference could be that these projects allow more open innovation and can more easily integrate external knowledge. It is also possible that business projects benefit less from larger clusters because there are already sufficient intra-firm knowledge flows, such that there are diminishing returns.

Alternative User Samples Lastly, we estimate the elasticity between cluster size and the number of commits for different user samples. Specifically, we calculate a user's share of commits among all commits, and restrict the sample to users in the upper 25 percent, upper 50 percent and upper 75 percent of the distribution of commits per user. More productive users (in terms of total commits per user) might benefit more from larger clusters.[28] Table 6 presents the estimates for the three subsamples of users, in decreasing order, and the baseline estimate in the final column.

All estimates are conditional on all fixed effects. The elasticity for the upper 25 percent users is the largest with 0.5233 (0.3014) and significant at the ten percent level. The coefficient for the upper 50 percent is 0.3227 (0.1607) and significant at the five percent level. For the upper 75 percent of users, the elasticity is also significant at the five percent level with 0.2930 (0.1312) and similar to the baseline estimate of 0.2775 (0.1253).

Users in the highest activity quartile might use GitHub for work reasons, hence a change in cluster size might increase their commits the most.

4.4 Mechanism

In this section we want to analyze, which projects are driving our results. There are several characteristics of the projects in our sample that may help to explain which projects mainly contribute to our baseline elasticity.

[28] Note that the results of this heterogeneity analysis have to be interpreted cautiously, as the sample split is, to some extent, based on the outcome, i.e., we examine the effects of cluster size within activity quartiles.

Table 6: Alternative User Samples

	Log(Commit)			
	Upper 25%	Upper 50%	Upper 75%	All
	(1)	(2)	(3)	(4)
Log(Size)	0.5233*	0.3227**	0.2930**	0.2775**
	(0.3014)	(0.1607)	(0.1312)	(0.1253)
Adjusted R^2	0.390	0.326	0.298	0.292
Observations	823,076	1,779,758	2,390,199	2,527,496

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression. Controls for city, time, language, city × language, language × time, user, city × technology, technology, city × time and project are included. Users are measured by their share of commits to all commits. Hence, users in the upper 25% sample cover the upper 25% of all commits by their commits.

First, about 50 percent of projects with at least two users committing to, are co-located projects, with all users stemming from the same city.[29] Splitting the sample of projects with at least two committers into projects where all users stem from one city vs. more geographically distributed projects in Table 7 shows that the effect for co-located projects with 0.2932 (0.2504) is larger than our baseline elasticity, but insignificant. For more distributed projects, the elasticity is very similar to our baseline elasticity with 0.2736 (0.1303) and significant at the five percent level. It seems that especially distributed projects drive our results as they make up a large share of observations.

Table 7: Co-located and More Distributed Projects

	Log(Commit)	
	Distributed	Co-located
	(1)	(2)
Log(Size)	0.2736**	0.2932
	(0.1303)	(0.2504)
Adjusted R^2	0.359	0.299
Observations	830,118	168,362

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression. The sample includes only projects with at least two users committing to and is split into co-located projects, whose members are all in the same city, and distributed projects, where at least one member is located in a different city from the other members. In all regressions, fixed effects for city, time, programming language, city × programming language, programming language × time, city × technology, technology, city × time, project and user are included.

[29] In 75% of all projects, i.e. including single projects, all users stem from one city. See Table 16 for more details.

Productivity increases seem also to depend on the team (i.e. project) size. When restricting the sample to projects with at most a total of 40 users committing to, the elasticity again resembles our baseline elasticity with 0.2599 (0.1236).[30] For larger projects, i.e. with more than 40 users, the effect almost triples to 0.8183 (0.3548) and is still significant at the five percent level. As Ayoubi et al. (2017) show, the probability of learning from team members is higher in larger teams. The larger elasticity for projects with more than 40 users may capture next to knowledge spillovers within clusters also knowledge spillovers within teams. In our sample, though, most projects are rather small as can be seen by the number of observations.[31]

Table 8: Small and Large Projects

	Log(Commit)	
	Small	Large
	(1)	(2)
Log(Size)	0.2599**	0.8183**
	(0.1236)	(0.3548)
Adjusted R^2	0.303	0.421
Observations	2,448,041	79,455

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression. For every project the total number of users committing to the project over the whole time was calculated. Small projects are projects with at most 40 total users, large projects have more than 40 users in total committing to. In all regressions, fixed effects for city, time, programming language, city × programming language, programming language × time, city × technology, technology, city × time, project and user are included.

A concern regarding our data is, that we do not observe the content of commits. Thus, the commits may be for file storage on GitHub and not necessarily commits for software development. Kalliamvakou et al. (2016) find that only about two thirds of projects on GitHub are for software development and a majority of projects are not set up for collaboration but rather individual projects. These projects also receive commits mainly by the project owner. However, this does not seem to be the case for our sample. Splitting the sample to commits to only others' projects vs. commits to the user's own project shows, that the effect is only significant for commits to others' projects. The commits to others' projects more likely are improvements to the projects and resemble productive output. This mitigates the concern about the commit content.

To sum up, team-wise smaller and more localized leisure projects are the main contributors to our estimate. For those projects a positive elasticity can be found between cluster size and productivity.

[30] We chose 40 as the cutoff for total users per project as Lima et al. (2014) found that geographically more distributed teams are observed mainly for projects with more than 40 users.

[31] Kalliamvakou et al. (2014) and Lima et al. (2014) also show that most projects on GitHub consist of small teams.

Table 9: Others and Own Projects

	Log(Commit)	
	Others	Own
	(1)	(2)
Log(Size)	0.3061**	0.0669
	(0.1494)	(0.1417)
Adjusted R^2	0.324	0.314
Observations	1,423,404	1,104,092

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression. Ownership of a project is determined if author id equals project owner id. In all regressions, fixed effects for city, time, programming language, city × programming language, programming language × time, city × technology, technology, city × time, project and user are included.

4.5 Instrumental Variable Estimates

To address endogeneity concerns potentially biasing the baseline estimates, we use an instrumental variable to isolate changes in local cluster size originating elsewhere by estimating equation (2). In this setting, unobserved time-varying productivity shocks on the city technology level that simultaneously affect user productivity and cluster size, and as a result, bias the estimated elasticity, should be removed.

For comparison, Table 20 in the Appendix presents the baseline model estimated as first differences. The sample consists of all users from the baseline estimates that commit to projects in two consecutive time intervals. In this case, the estimated coefficients are positive, but statistically insignificant and small in magnitude. The estimate reflects the contemporaneous change in productivity with a change in cluster size. The coefficient of cluster size is smaller than the baseline elasticity and positive with 0.0113 (0.0100) in column four, conditional on time, city, user, programming language and programming language × time fixed effects.[32] The smaller and positive estimate in comparison to the baseline estimate with 0.2775 (0.1253) is partly due to its representation of the contemporaneous effect of cluster size on productivity, excluding effects of knowledge spillovers that increase productivity with a delay. Furthermore, in first difference models, measurement errors are magnified (Griliches and Hausman, 1986). Cluster size is likely measured imprecisely to a certain extent, due to wrong self-stated user locations. This might cause the estimate to become smaller. The positive coefficient suggests that a contemporaneous change in cluster size positively affects productivity.

Instrumenting changes in local cluster size by changes in cluster size originating elsewhere, the estimate conditional on covariates, becomes 0.19829 (0.09711) and significant at the five percent level as

[32]Not all covariates from the baseline model are included. In first difference models, time-invariant factors are canceled out. As the bias due to unobservable city programming language productivity shocks should be removed by the instrument, controlling for city × programming language might be too conservative.

Table 10: 2SLS Estimates

	Δ Log(Commit) (1)	Δ Log(Commit) (2)	Δ Log(Commit) (3)	Δ Log(Commit) (4)
First Stage	-0.00001***	-0.00001***	-0.00001***	-0.00001***
	(0.00000)	(0.00000)	(0.00000)	(0.00000)
Δ Log(Size)	0.20336	0.29913***	0.29436***	0.19829**
	(0.19268)	(0.08786)	(0.08690)	(0.09711)
Fixed-effects				
Time	Yes	Yes	Yes	Yes
City	Yes	Yes	Yes	Yes
User		Yes	Yes	Yes
Language			Yes	Yes
Language x Time				Yes
Observations	500,665	500,665	500,665	500,665
F-test (1st stage)	466.53	1,317.96	1,336.73	1,479.94

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city. Every column presents a regression. The sample consists of commits to projects, that receive commits in two consecutive time intervals. The dependant variable is the change in the log of commits to a project between two consecutive time intervals. The model estimated is equation (3).

presented in Table 10, column four. The first stage in column four is significant at the one percent level with an estimate of -0.00001 (0.00000) conditional on all covariates. The smaller effect in the second stage compared to the OLS estimate stems from the fact that the instrument corrects for measurement error and endogeneity.

Changes in cluster size elsewhere are a good predictor of local cluster size even though their magnitude is relatively small. The instrument is very strong with an F-test of 1,479.94. The negative coefficient of the instrument suggests, that changes in other projects outside the local cluster are associated with a decrease in local cluster size. A possible explanation may be that users move to larger clusters. If clusters elsewhere increase, captured by increased users committing to projects outside the local cluster, the local cluster would decrease. Overall, the instrumental variable regressions confirm the existence of positive agglomeration effects, even though OLS regressions may somewhat overestimate their magnitude.

4.6 Robustness

Alternative User Samples In the main analysis, only users are included that commit in all time intervals. They represent the more active users and as a result, the absolute elasticity may be the largest for them. Furthermore, as Casalnuovo et al. (2015) show, productivity increases are the largest in collaboration for users with greater knowledge in a programming language and, thus, technology. Users with commits in all time intervals likely have a good level of knowledge in a technology which may lead to

having the largest elasticity between productivity and cluster size. Therefore, we loosen the restriction on the time intervals with non-zero commits per user. If the assumption is true, that the users with non-zero commits in all time intervals are the most active, we would expect the absolute elasticity between cluster size and productivity decrease by decreasing the number of time intervals with non-zero commits. Table 11 presents the regression results for different subsamples with all covariates from the baseline model included.

Table 11: Different Lengths of Observation Period

	Log(Commit)				
	1	2	3	4	5
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1989*	0.1986*	0.1894*	0.1911*	0.2104**
	(0.1082)	(0.1076)	(0.1029)	(0.1018)	(0.1005)
Adjusted R^2	0.173	0.180	0.227	0.249	0.261
Observations	6,363,687	6,320,343	5,295,433	4,636,989	4,053,452

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Table 12: Different Lengths of Observation Period - Continued

	Log(Commit)				
	6	7	8	9	10
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.2209**	0.2175**	0.2432**	0.2524**	0.2775**
	(0.1027)	(0.1043)	(0.1081)	(0.1127)	(0.1255)
Adjusted R^2	0.268	0.273	0.278	0.284	0.292
Observations	3,722,638	3,470,489	3,197,671	2,961,254	2,527,496

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard Errors are clustered by city x technology. Every column presents a regression of equation 1. In column 1, users are included that in at least one time interval had commits. In column 2, users are included that commit in at least two time intervals, and so on.

In the first column, users are included that commit in at least one time interval. In column two, the sample consists of users that commit in at least two consecutive time intervals. Column three shows the regression results of users with commits in at least three consecutive time intervals. This way, the column number represents the number of consecutive time intervals with non-zero commits per user. The elasticity is significant at the ten percent level in the first four columns and decreases from 0.1989 (0.1082) to 0.1911 (0.1018). This indicates that for the sample of users with non-zero commits between at least one to at least four time intervals contains users for which a smaller relationship between cluster size

and productivity is found. One reason might be, due to their lower activity on GitHub, not all productivity changes can be observed.

The coefficients increase in size in the following columns five to ten, as well as becoming significant at the five percent level. The elasticity in column ten, which presents the baseline estimate, is the largest in size with 0.2775 (0.1255). Hence, the assumption of the largest elasticity for the most active users is confirmed by these results.

It is further worth noting that the results may also reflect a possible bias towards zero. The results represent the intensive margin, i.e., an increase in commits with an increase in cluster size, given a user commits. In the case of zero commits, we do not observe the user's productivity. If a larger cluster also affects the probability to commit (extensive margin), and given this effect goes in the same direction as the effect on the intensive margin, the estimates would be biased towards zero. In column ten, only users with non-zero commits are included, and hence this concern should be mitigated. The estimate is the largest in size, which supports the hypothesis of a bias towards zero in the other columns.

Alternative Measure of Cluster Size We now turn to an alternative measure of cluster size. In the previous regressions, clusters were defined by the share of all local users, excluding the focal user, in a technology over all users in a technology in a time interval, irrespective of being active in that time interval or not. For robustness we calculate cluster size only based on active users, i.e. with commits in the respective technology.

Table 13 presents the coefficients for the elasticity between cluster size, measured by active users, and commits. In the first column, with controls for city, time, programming language, technology, project and user, the estimate is positive with 0.0918 (0.0748) but insignificant. After adding further controls for city $\times$ technology, city $\times$ programming language, programming language $\times$ time, and city $\times$ time, the estimate increases to 0.1622 (column five; 0.0684) and is significant at the five percent level. The results suggest that a ten percent increase in the share of local active users to all active users in a technology increases the number of commits in that technology by 1.62 percent. This implies a significant increase in the number of commits as a result of more local active users, i.e. it seems that they positively influence a user's activity. In comparison to the baseline estimate with 0.2775 (0.1255), the elasticity becomes smaller when restricting cluster calculation to only active users.

The results confirm our baseline estimates of a positive elasticity between cluster size and user activity.

5 Conclusion

Concentration into one or few large clusters is observed in many industries in the United States. This is especially the case in the software industry, with Silicon Valley and a small number of other agglom-

Table 13: Active Users

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Active Users)	0.0918	0.0726	0.1009*	0.0995*	0.1662**
	(0.0748)	(0.0468)	(0.0561)	(0.0567)	(0.0684)
Fixed-effects					
City	Yes	Yes	Yes	Yes	Yes
Time	Yes	Yes	Yes	Yes	Yes
Language	Yes	Yes	Yes	Yes	Yes
Technology	Yes	Yes	Yes	Yes	Yes
Project	Yes	Yes	Yes	Yes	Yes
User	Yes	Yes	Yes	Yes	Yes
Language x Time		Yes	Yes	Yes	Yes
City x Technology			Yes	Yes	Yes
City x Language				Yes	Yes
City x Time					Yes
Adjusted R^2	0.287	0.290	0.291	0.291	0.292
Observations	2,526,979	2,526,979	2,526,979	2,526,979	2,526,979

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression.

erations having outsize importance, even though software is both used and developed across industries and regions. Why do companies and software engineers choose these expensive locations? For a selected group of top inventors, Moretti (2021) shows positive effects of cluster size on innovation. We contribute by studying a larger segment of the labor market, software engineers, using fine-grained data from online collaboration on open source projects from GitHub.

We find a significant elasticity of 0.2775 (0.1253) between productivity and cluster size for GitHub users conditional on several controls. The estimated effect is larger for commits to projects in the top decile of received stars with 0.3239 (0.1462). Projects with more stars likely demand higher quality of their receiving commits and the results, thus, imply an increase in the quality of commits with an increase in cluster size. The heterogeneity analysis showed, that especially older projects and projects with a smaller share of commits made during business hours benefit from increases in cluster size.

The mechanisms underlying the knowledge spillovers, e.g., task specialization or training, could be the focus of future research. With an increase in cluster size, the task distribution might change, e.g., every project member only uses one programming language.

Unfortunately, we are only able to observe commits to public projects. If programmers in larger clusters tended to shift their activity towards private closed source projects, our estimates would be a lower bound elasticity between cluster size and productivity. Additionally, we do not analyze the commit

content, and rely on the assumption that the value of commits does not systematically change with cluster size. Lastly, cluster size may be measured imprecisely due to not updated user profiles and, thus, wrong assignment of users to a cluster.

With these limitations in mind, our results suggest that productivity spillover effects and cluster size play an important role in open source software development as well. They are a valuable input source for firms and, thus, fostering knowledge creation would further increase the benefits from integrating open source software. Even in times of increasing working from home, cities and urban density keep playing an important role in the diffusion of knowledge. Our results on agglomeration effects among software engineers are also relevant for policy choices. For example, policies focused on tech startups and digitization in general may be more successful, if they focus on spreading applications of new technologies across the country, whereas the development of more novel software may be more productive in existing and dense clusters.

References

Andersson, M., J. Klaesson, and J. P. Larsson (2014). The sources of the urban wage premium by worker skills: Spatial sorting or agglomeration economies? *Papers in Regional Science 93*(4), 727–747.

Andersson, R., J. M. Quigley, and M. Wilhelmsson (2009). Urbanization, productivity, and innovation: Evidence from investment in higher education. *Journal of Urban Economics 66*(1), 2–15.

Atkin, D., M. K. Chen, and A. Popov (2022). The returns to face-to-face interactions: Knowledge spillovers in silicon valley. Technical report, National Bureau of Economic Research.

Au, C.-C. and J. V. Henderson (2006). Are chinese cities too small? *The Review of Economic Studies 73*(3), 549–576.

Ayoubi, C., M. Pezzoni, and F. Visentin (2017). At the origins of learning: Absorbing knowledge flows from within the team. *Journal of Economic Behavior & Organization 134*, 374–387.

Azoulay, P., J. S. Graff Zivin, and J. Wang (2010). Superstar extinction. *The Quarterly Journal of Economics 125*(2), 549–589.

Becker, R. A. and A. R. Wilks (2018). Package 'maps'. `https://cran.r-project.org/web/packages/maps/maps.pdf`. Accessed: 2021-05-11.

Belenzon, S. and M. Schankerman (2015). Motivation and sorting of human capital in open innovation. *Strategic Management Journal 36*(6), 795–820.

Bell, A., R. Chetty, X. Jaravel, N. Petkova, and J. V. Reenen (2019). Who becomes an inventor in America? The importance of exposure to innovation. *The Quarterly Journal of Economics 134*(2), 647–713.

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

Carlino, G. and W. R. Kerr (2015). Agglomeration and innovation. *Handbook of Regional and Urban Economics 5*, 349–404.

Carlino, G. A., S. Chatterjee, and R. M. Hunt (2007). Urban density and the rate of invention. *Journal of Urban Economics 61*(3), 389–419.

Carreteiro, P., J. B. d. Vasconcelos, A. Barão, and Á. Rocha (2016). A knowledge management approach for software engineering projects development. In *New advances in information systems and technologies*, pp. 59–68. Springer.

Casalnuovo, C., B. Vasilescu, P. Devanbu, and V. 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*, pp. 817–828.

Catalini, C. (2018). Microgeography and the direction of inventive activity. *Management Science 64*(9), 4348–4364.

Charlot, S. and G. Duranton (2004). Communication externalities in cities. *Journal of Urban Economics 56*(3), 581–613.

Cohen, J. E. and M. A. Lemley (2001). Patent scope and innovation in the software industry. *Calif. L. Rev. 89*, 1.

Combes, P.-P., G. Duranton, L. Gobillon, and S. Roux (2010). Estimating agglomeration economies with history, geology, and worker effects. In *Agglomeration economics*, pp. 15–66. University of Chicago Press.

Combes, P.-P. and L. Gobillon (2015). The empirics of agglomeration economies. In *Handbook of regional and urban economics*, Volume 5, pp. 247–348. Elsevier.

Cornelissen, T., C. Dustmann, and U. Schönberg (2017). Peer effects in the workplace. *American Economic Review 107*(2), 425–56.

Duranton, G. and D. Puga (2001). Nursery cities: Urban diversity, process innovation, and the life cycle of products. *American Economic Review 91*(5), 1454–1477.

Fackler, T. A., Y. Giesing, and N. Laurentsyeva (2020). Knowledge remittances: Does emigration foster innovation? *Research policy 49*(9), 103863.

Giddings, F. H. (1890). Principles of economics. by alfred marshall, professor of political economy in the university of cambridge. macmillan & co., london and new york, 1890. vol. i, pp. xxviii, 754. *The ANNALS of the American Academy of Political and Social Science 1*(2), 332–337.

GIT (2021). `https://git-scm.com/`. Accessed: 2021-05-31.

Glaeser, E. L. (1999). Learning in cities. *Journal of urban Economics 46*(2), 254–277.

Gousios, G. (2013). The ghtorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, Piscataway, NJ, USA, pp. 233–236. IEEE Press.

Griliches, Z. and J. A. Hausman (1986). Errors in variables in panel data. *Journal of econometrics 31*(1), 93–118.

Hergueux, J. and N. Jacquemet (2015). Social preferences in the online laboratory: A randomized experiment. *Experimental Economics 18*, 251–283.

Hindle, A., D. M. German, and R. Holt (2008). What do large commits tell us? a taxonomical study of large commits. In *Proceedings of the 2008 international working conference on Mining software repositories*, pp. 99–108.

Jaffe, A. B., M. Trajtenberg, and R. Henderson (1993). Geographic localization of knowledge spillovers as evidenced by patent citations. *the Quarterly journal of Economics 108*(3), 577–598.

Kalliamvakou, E., G. Gousios, K. Blincoe, L. Singer, D. German, and D. Damian (2014). The promises and perils of mining GitHub. In *Proceedings of the 11th working conference on mining software repositories*, pp. 92–101.

Kalliamvakou, E., G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian (2016). An in-depth study of the promises and perils of mining github. *Empirical Software Engineering 21*(5), 2035–2071.

Laurentsyeva, N. (2019). From friends to foes: National identity and collaboration in diverse teams. Technical report, Discussion Paper.

Lima, A., L. Rossi, and M. Musolesi (2014). Coding together at scale: Github as a collaborative social network. *Proceedings of the international AAAI conference on web and social media 8*(1), 295–304.

Mas, A. and E. Moretti (2009). Peers at work. *American Economic Review 99*(1), 112–45.

McDermott, G. R. and B. Hansen (2021). Labor reallocation and remote work during covid-19: Real-time evidence from github. Technical report, National Bureau of Economic Research.

Moretti, E. (2021). The effect of high-tech clusters on the productivity of top inventors. *American Economic Review 111*(10), 3328–75.

Nagle, F. (2019). Open source software and firm productivity. *Management Science 65*(3), 1191–1215.

Prana, G. A. A., D. Ford, A. Rastogi, D. Lo, R. Purandare, and N. Nagappan (2021). Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in oss. *IEEE Transactions on Software Engineering 48*(9), 3394–3409.

Riehle, D. (2012). The single-vendor commercial open course business model. *Information Systems and e-Business Management 10*(1), 5–17.

Rosenthal, S. S. and W. C. Strange (2020). How close is close? The spatial reach of agglomeration economies. *Journal of Economic Perspectives 34*(3), 27–49.

Schumpeter, J. A. (1939). *Business cycles*, Volume 1. McGraw-Hill New York.

Simplemaps (2021). United states cities database. `https://simplemaps.com/data/us-cities`. Accessed: 2021-05-10.

Stack Overflow (2020). Stack overflow developer survey 2020. `https://insights.stackoverflow.com/survey/2020#overview`. Accessed: 2022-05-30.

Wachs, J., M. Nitecki, W. Schueller, and A. Polleres (2022). The geography of open source software: Evidence from github. *Technological Forecasting and Social Change 176*, 121478.

A Appendix

A.1 Tables

Table 14: Summary Statistics Commits by Programming Languages

Language	Min.	Median	Mean	Max.	Projects	N	Commits	Share
C	1	20	347.72	60,823	74,069	8,736	3,037,651	6.26%
C#	1	21	322.57	18,116	37,097	4,183	1,349,295	2.78%
C++	1	24	375.46	56,647	67,807	8,806	3,306,298	6.81%
CSS	1	34	142.08	225,318	55,866	13,590	1,930,912	3.98%
Go	1	20	287.12	25,176	60,595	6,286	1,804,832	3.72%
HTML	1	38	180.66	72,061	112,391	15,814	2,857,032	5.89%
Java	1	26	386.65	221,308	101,568	9,769	3,777,152	7.78%
JavaScript	1	121	497.95	172,663	354,177	17,766	8,846,592	18.23%
Jupyter Notebook	1	17	114.44	9,212	13,020	2,940	336,468	0.69%
Objective-C	1	10	116.23	9,075	18,664	3,486	405,164	0.83%
PHP	1	24	355.45	218,260	69,086	7,165	2,546,832	5.25%
Python	1	61	457.28	29,471	185,052	14,645	6,696,895	13.8%
R	1	26	404.04	73,039	18,555	1,696	685,248	1.41%
Ruby	1	34	339.01	48,734	135,510	10,406	3,527,781	7.27%
Rust	1	19	221.37	41,287	19,339	2,677	592,599	1.22%
Shell	1	26	453.35	3,761,123	64,945	12,533	5,681,893	11.71%
Swift	1	15	151.80	30,077	12,739	2,035	308,920	0.64%
TypeScript	1	15	139.70	20,600	30,123	6,011	839,763	1.73%

Table 15: Summary Statistics Commits by Technologies

Technology	Min.	Median	Mean	Max.	Projects	N	Commits	Share
1	1	326	911.59	235,565	658,740	20,152	18,370,426	37.85%
2	1	136	842.12	3,761,126	342,167	18,056	15,205,336	31.33%
3	1	34	339.01	48,734	135,510	10,406	3,527,781	7.27%
4	1	33	392.49	221,308	132,971	11,443	4,491,236	9.25%
5	1	44	565.74	60,884	161,215	12,261	6,936,548	14.29%

Table 17: Largest Clusters for Technologies in 202103 Snapshot

	Size
1	
San Jose-San Francisco-Oakland, CA	0.10638
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.08784
Seattle-Tacoma-Olympia, WA	0.05366

Los Angeles-Long Beach-Riverside, CA	0.04403
Indianapolis-Anderson-Columbus, IN	0.04387
Toronto	0.03732
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03484
Boston-Worcester-Manchester, MA-NH	0.03233
Chicago-Naperville-Michigan City, IL-IN-WI	0.03200
Dallas-Fort Worth, TX	0.02390
2	
San Jose-San Francisco-Oakland, CA	0.13441
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.09031
Seattle-Tacoma-Olympia, WA	0.05627
Boston-Worcester-Manchester, MA-NH	0.04299
Los Angeles-Long Beach-Riverside, CA	0.04095
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.04039
Toronto	0.03375
Indianapolis-Anderson-Columbus, IN	0.03250
Chicago-Naperville-Michigan City, IL-IN-WI	0.03073
Denver-Aurora-Boulder, CO	0.02385
3	
San Jose-San Francisco-Oakland, CA	0.14154
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.11862
Seattle-Tacoma-Olympia, WA	0.04749
Chicago-Naperville-Michigan City, IL-IN-WI	0.04181
Los Angeles-Long Beach-Riverside, CA	0.04000
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03808
Boston-Worcester-Manchester, MA-NH	0.03748
Denver-Aurora-Boulder, CO	0.03744
Toronto	0.03028
Indianapolis-Anderson-Columbus, IN	0.02649
4	
San Jose-San Francisco-Oakland, CA	0.13405
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.08113
Indianapolis-Anderson-Columbus, IN	0.05250
Seattle-Tacoma-Olympia, WA	0.05092
Los Angeles-Long Beach-Riverside, CA	0.04059

Toronto	0.03623
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03382
Boston-Worcester-Manchester, MA-NH	0.03259
Chicago-Naperville-Michigan City, IL-IN-WI	0.03208
Dallas-Fort Worth, TX	0.02942
5	
San Jose-San Francisco-Oakland, CA	0.13050
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.06623
Seattle-Tacoma-Olympia, WA	0.05999
Los Angeles-Long Beach-Riverside, CA	0.04048
Indianapolis-Anderson-Columbus, IN	0.03647
Boston-Worcester-Manchester, MA-NH	0.03572
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03342
Toronto	0.02954
Dallas-Fort Worth, TX	0.02884
Dayton-Springfield-Greenville, OH	0.02735

Table 16: Summary Statistics of Commits, Projects and Users

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Length User Observed	10	10	10	10.00	10	10
Commits per User	25	521	1,059	2,298.32	2,286	3,767,493
Commit per Project per Snapshot	1	1	3	19.20	10	1,298,112
Stars per Project	0	0	0	84.75	2	259,118
Programming Language per City	1	12	17	14.36	18	18
Programming Language per City per Snapshot	1	6	11	10.26	15	18
Technology per City	1	5	5	4.57	5	5
Technology per City per Snapshot	1	3	4	3.64	5	5
Programming Language per User	1	5	7	7.03	9	18
Programming Language per User per Snapshot	1	2	3	3.38	4	17
Technology per User	1	3	3	3.42	4	5
Technology per User per Snapshot	1	1	2	2.25	3	5
Own Project	0	0	0	0.42	1	1
Business Share	0	0	1	0.62	1	1
Weekend Share	0	0	0	0.19	0	1
Out of Hour Share	0	0	0	0.31	0	1
Local Share	0	1	1	0.90	1	1
Users per Project	1	1	1	1.68	1	2,381
Project Age (in Years)	0	3	5	4.96	7	13

Table 18: Summary Statistics - Clusters

Technology	10th Perc.	Median	90th Perc.	Max.
1	0.00002	0.00045	0.01054	0.10638
2	0.00001	0.00048	0.01042	0.13441
3	0.00003	0.00055	0.01032	0.14154
4	0.00002	0.00048	0.00992	0.13405
5	0.00002	0.00060	0.01019	0.13050

Table 19: Baseline Estimates - Excluding Projects with large Commits and large Projects

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1006	0.0786	0.1626*	0.1584*	0.2771**
	(0.1001)	(0.0635)	(0.0941)	(0.0954)	(0.1168)
Fixed-effects					
City	Yes	Yes	Yes	Yes	Yes
Time	Yes	Yes	Yes	Yes	Yes
Language	Yes	Yes	Yes	Yes	Yes
Technology	Yes	Yes	Yes	Yes	Yes
Project	Yes	Yes	Yes	Yes	Yes
User	Yes	Yes	Yes	Yes	Yes
Language x Time		Yes	Yes	Yes	Yes
City x Technology			Yes	Yes	Yes
City x Language				Yes	Yes
City x Time					Yes
Adjusted R^2	0.256	0.260	0.260	0.260	0.261
Observations	2,382,259	2,382,259	2,382,259	2,382,259	2,382,259

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city x technology. Every column presents a regression. Sample includes only projects with less than 40 users committing to and commits to projects less than 100.

Table 20: First Differences Estimates

	Δ Log(Commit)			
	(1)	(2)	(3)	(4)
Δ Log(Size)	-0.0017	0.0123	0.0124	0.0113
	(0.0108)	(0.0097)	(0.0096)	(0.0100)
Fixed-effects				
Time	Yes	Yes	Yes	Yes
City	Yes	Yes	Yes	Yes
User		Yes	Yes	Yes
Language			Yes	Yes
Language x Time				Yes
Adjusted R^2	0.082	0.086	0.086	0.087
Observations	500,665	500,665	500,665	500,665

*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1.*

Notes: Standard errors are clustered by city Every column presents a regression. The sample consists of commits to projects, that receive commits in two consecutive time intervals. The dependant variable is the change in the log of commits to a project between two consecutive time intervals. The model estimated is equation (3).