Productivity Spillovers among Knowledge Workers in Agglomerations: Evidence from GitHub

Lena Abou El-Komboz[*], Thomas Fackler[†]

August 13, 2022

Abstract

How much do high-tech workers benefit from being physically close to each other? On the one hand, software engineers' work could almost entirely be done remotely. On the other hand there is an extensive literature on clustering and agglomeration effects on innovative behavior. Using a large data set with over 11 million observations covering the years 2015 to 2021 from the open source platform GitHub, we relate cluster size to a user's productivity. Our findings suggest that high-tech workers benefit from physical proximity to a large number of other workers in their field. In further analyses, we implement an event-study design and study and heterogeneities by cluster size, project productivity and other project characteristics.

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

[*]ifo Institute and LMU Munich, Poschingerstraße 5, 81679 München, abou-el-komboz@ifo.de.
[†]ifo Institute, LMU Munich and CESifo, Poschingerstraße 5, 81679 München, fackler@ifo.de.

1 Introduction

During the COVID-19 pandemic, a large shift from office work to work from home was observed. In the past, workers had disadvantages when working from home such as lower wages in comparison to their coworkers working on-site (Emanuel and Harrington, 2020). Employers were of the opinion that productivity decreases when an employee was not physically present at the workplace. Now, as work from home is more common, these prejudices may decrease (Emanuel and Harrington, 2020).[1]

This shift to remote work prompts various follow-up questions regarding infrastructure and urban utilization: What to do with the now empty offices? Do they become redundant, if employees work remotely? Will employees move to the countryside to benefit from lower rents? In that case, cities as we know them will change and urban density may decrease.

Research suggests that this is unlikely: Cities with higher densities have benefits beyond the smaller distance of employees to their workplace. Urban density was found to positively affect wages as well as worker and firm productivity. One reason seems to be the better diffusion of knowledge with physical proximity. Knowledge spillovers occur between workers, i.e., workers benefit from the differing skills of their coworkers and by learning from them, they gain new skills and their productivity increases (Cornelissen et al., 2017). Especially in the innovative sector, strong clustering is observed because workers tend to locate near each other (Demsas, 2021).

Agglomeration economics captures this process. It conceptualizes the effects on workers and firms with an increase in urban density (Combes and Gobillon, 2015). Several findings suggest that proximity plays a big role for productivity spillovers. The effects rapidly decay with an increase in distance to each other (Baum-Snow et al., 2020).[2] Innovation clusters seem to locate near universities to benefit from the decreased costs of communication and increased knowledge flow (Andersson et al., 2009).

Patent data is often used to measure innovative activity and the effects of agglomeration on it. However, the data likely does not capture all productivity gains from an increase in urban density (Carlino and Kerr, 2015).

To capture more fine-grained productivity effects, we use an alternative data source. GitHub[3] is the world's biggest open source platform. Software developers collaborate on projects through the platform. The work on public projects is observable by everyone. A user in a larger cluster

[1]Emanuel and Harrington (2020) find, based on natural experiments, that before the COVID-19 pandemic workers were adversely selected into remote work. Less productive workers were more likely to work from home and wages for remote work were lower. Now in the COVID-19 pandemic, productivity gains of working from home for workers are observed. The researchers conclude that, before the COVID-19 pandemic, too few employees worked from home given the productivity gains they would experience with a change of workplace.

[2]Baum-Snow et al. (2020) find that firm and productivity spillovers fully decline within 250 meters. They also show, that these spillover effects are higher for higher quality firms.

[3]`https://github.com/`

might also benefit from knowledge spillovers and become more active on the platform. So even smaller shifts in productivity with an increase in cluster size are likely captured by the data.

Open source software plays an important role for firms. Firms that incorporate open source software experience an increase in value-added productivity. These platforms are places where a lot of productive output is generated (Nagle, 2019).[4]

It is also part of the high-tech sector[5] and based on prior research (Casalnuovo et al., 2015), clusters should occur and matter as well for GitHub users' productivity.

To study agglomeration effects on the productivity of GitHub users in the USA and Canada, we build on the empirical approach by Moretti (2021). We use the exogenous variation in cluster size originating from users moving across cities, to estimate the impact of an increase in cluster size on the users' productivity. This can be via affecting the output's quantity as well as quality. Furthermore, is there heterogeneity in the effects, i.e., does the effect differ on several aspects such as cluster size or productivity level of a user? The findings are important to foster open source activity and maximize the gains from it, but also to improve our understanding of agglomeration effects among knowledge workers more generally.

A concern when estimating agglomeration effects on productivity are simultaneity and unobserved productivity shocks. Therefore, we implement an event-study design based on movers. The setting allows us to study the dynamic response of a user to a move. We find positive productivity gains with the move, whereas lagged values of cluster size do not affect current productivity.

To begin with, we explain the research question in more detail and give an overview of the existing literature on innovation and agglomeration economics in section two. Then, in section three, the theoretical interplay of knowledge spillovers, productivity, and urban density is described. Thereafter, the empirical framework is presented in section four. First, GitHub is explained in detail, as well as the data gathering. Second, the data is described and possible limitations of it are discussed. Third, the estimation strategy to identify agglomeration effects on GitHub users' productivity is explained. The findings of the empirical analysis are presented in the following section five. Finally, we conclude in section six.

2 Motivation

In this section, the research question is presented to analyze the effects of cluster size on the productivity of GitHub users. Furthermore an overview of the existing literature on the effects of exposure to innovation and productivity as well as agglomeration effects and productivity is given.

[4]A firm incorporating one percent more free open source software raises the value-added productivity by 0.002 to 0.008 percent (Nagle, 2019)

[5]In line with Moretti (2021), we use the term high-tech to describe any firm generating innovations.

2.1 Research Question

Workers and firms relying on knowledge-intensive tasks spatially cluster more than, e.g., manufacturing employment (Carlino and Kerr, 2015; Audretsch and Feldmann, 1996). A higher urban density tends to positively affect the number of patents in that area (Carlino et al., 2007). If an inventor moves to such an area, her productivity, measured by e.g. number of patents, seems to increase (Moretti, 2021). This hints at productivity spillover effects of local inventors affecting the productivity of the moving inventor.

The increase in innovative activity as a result of a relocation to a denser area might be due to an increase in collaborations with local inventors (Catalini, 2018). Specifically, active exposure to innovation matters: For instance after an inventor dies, the positive exposure effect on productivity among co-authors of the departed inventor likely decreases (Azoulay et al., 2010).

The majority of studies on exposure to innovation and agglomeration effects measures productivity by patents. Yet, as studies suggest, using patent data to measure innovative activity has its downsides (Carlino and Kerr, 2015). Not all patents possess the same value. Rather, it seems that there are some very valuable patents containing important innovations, whereas a large number of patents are less valuable. The latter ones might be too specific or were filed for legal reasons (Carlino and Kerr, 2015).

Another shortcoming of patent data to analyze innovation is its lack of representativeness of innovation. A patent is the first step of the innovative process, namely, the invention, but not every innovative idea is patented (Carlino and Kerr, 2015). To patent or not is also field specific and whether innovations are commonly patented or not varies across domains (Cohen et al., 2000). Therefore, patent data does not capture all innovative activity.

In conclusion, measuring innovation based on patent data has its disadvantages. However, research finds positive cluster effects on patent recording and citation, suggesting spillover effects between innovators' productivity exist (Carlino and Kerr, 2015).

An alternative way to analyze these cluster effects, at least in the high-tech sector, is using GitHub data to measure productivity. GitHub is a code hosting platform based on the `git` version control system (Kalliamvakou et al., 2014; GIT, 2021). Developers can upload (*push*) their code changes onto the platform into a project repository. Others can then download (*pull*) the project and work on the code and modify it. Any code modification is called a *commit* (Laurentsyeva, 2019). The progress of a project depends heavily on the number of commits and hence a commit can be seen as a productivity unit (McDonald and Goggins, 2013). As a user, the main work interactions with other developers are commits. On the user profile pages, the commits by project are summarized to show how active a user is (Laurentsyeva, 2019).

Open Source platforms gain in importance, as for example, Microsoft even bought GitHub in 2018 for $7.5 billion (Microsoft, 2021). This shows, that for software firms these platforms are important input for their work and they are willing to invest large amounts of money to incorporate them.

The field of software programming is rapidly expanding and emerging. A software evolves over time and new components are added or old components removed. Patents in this sector likely only capture the lower bound of effects on productivity as well as innovative activity, because patent filing is a process that can take several years (Cohen et al., 2000).

In comparison to patent data, commits are not necessarily the first stage of innovation, but rather can also represent later stages in the innovative process. A project goes through several stages of development such as building, integration testing or system testing. With every commit, a project further evolves and becomes more innovative (Vasilescu et al., 2015). For a public GitHub project, these steps are observable by everyone. Furthermore, the values of commits are possibly more normally distributed in comparison to the high skewness of patents' values (Carlino and Kerr, 2015).

Even though programming is done on the computer and no real-world contact is necessary, interactions and knowledge diffusion in the real world might positively affect the advances of a project through an increase in commits or an increase in the quality of commits. Patent citations are very localized and the number of (local) patent citations also increases with cluster size, meaning in larger clusters, in comparison to smaller clusters, more citations are observed (Jaffe et al., 1993). The increase in local patent citations with cluster size points to better knowledge diffusion in bigger clusters caused by a more efficient way of knowledge flow (Moretti, 2021). GitHub users likely benefit from the increased knowledge flow as well.

Building on these advantages of GitHub, we use the empirical approach by Moretti (2021) to analyze cluster effects using GitHub data. Specifically, we take the number of commits of a developer as a measure of her productivity. Based on that, we estimate how the cluster size of a programming language affects her productivity in that programming language after a move to a larger cluster. With the data on hand, we find that cluster size positively affects a user's productivity as well as the quality of a user's output. The effects may differ on several aspects, e.g., the level of user or project productivity, or cluster size. So a heterogeneity analysis of the effects is carried out as well. The effects seem to be larger for more productive projects and users, and larger cluster. On the other hand, more skilled users, proxied by their number of followers, gain less from larger clusters than less-skilled users.

To test our results from the OLS regression using all variation, we estimate a dynamic response of productivity to a change in cluster size by including leads and lags of cluster size additionally to current cluster size. Productivity gains occur with a move and do not precede it. The first lead seems to affect current productivity, this can either be due to our data generating process or may suggest that productivity increases not only with the move but also in the next time interval. The findings support our main results of productivity gains with an increase in cluster size.

Cluster effects on productivity, as estimated by Moretti (2021) using patent data, are observable with GitHub data as well. After a user moves to a larger cluster, she might start to interact with local users. With larger cluster sizes, more users of a given cluster reside in a city and the relocated inventor can interact with a greater number of users in her cluster and collaborate with them. Thus, she might start contributing more to projects on GitHub.

Commits as a proxy for productivity to analyze agglomeration effects has not been tried in the literature. As research suggests, agglomeration effects are quite profound and even more so for skilled workers and knowledge-intensive tasks and, as shown, occur for GitHub users as well (Combes et al., 2010; Carlino et al., 2007).

On GitHub, users are arguably more skilled as it requires a certain level of knowledge to modify existing code. Based on this research design, it is possible to analyze even smaller shifts in productivity due to changes in cluster size. Not every innovative idea is patented and, thus, some productive developments caused by agglomeration effects are likely not observed. Every progress in a public GitHub project measured by a commit is observed and productivity gains of a move to a larger high-tech cluster may be more accurately captured.

Further understanding of agglomeration effects are important as governments attract companies to foster the development of high-tech clusters (Moretti, 2021). The reason lies in the fact that cities with large high-tech clusters tend to have higher mean wages and mean incomes in comparison to cities with less high-tech clusters. High-tech firms such as Amazon or Tesla are offered a large amount of subsidies to locate an establishment in a city. Obtaining more detailed knowledge on even smaller shifts in productivity as a result of agglomeration and collaboration effects is important to understand whether policymakers' hopes when attracting establishments are justified (Moretti, 2021). In particular, our study complements research on inventors and suggests that positive agglomeration effects are not limited to R&D, but extend to software engineering.

GitHub offers a perfect place to tackle this question due to the fine-grained data availability of interactions on the platform. Data on the history of interactions on GitHub as well as data on the users are provided by GitHub Torrent (Gousios, 2013) and are publicly accessible. Furthermore, GitHub contains integrated social features because it was constructed as an online platform for collaboration (Laurentsyeva, 2019). Therefore, it offers a perfect environment to study productivity spillover effects as they seem to be strongest in collaborations (Azoulay et al., 2010).

2.2 Overview of the Existing Literature

The main body of the innovation literature measures innovative activity with patents (Carlino and Kerr, 2015). The advantage of this approach lies in the data availability of patents. After its digitization in the late 1990s an increasing share of researchers measures productivity with patent data. The scope of this literature varies, among others, from the effects of exposure to innovation on productivity (Bell et al., 2019) to agglomeration effects on productivity (Moretti, 2021).

2.2.1 Effects of Exposure to Innovators and Productivity

Bell et al. (2019) analyzed the determinants of becoming an inventor by using patent data. They find that exposure to other inventors and network effects play an important role for children patenting later on in life. These effects seem to be gender- and technology class-specific, e.g., girls, which are exposed to women inventing in a specific field in their childhood, are more likely to patent in that field when grown up (Bell et al., 2019).[6] The findings of technology class-specific exposure effects show, that not the total number of inventors, or city size, matters for innovation but the field-specific cluster size.

Exposure to other inventors does not only affect innovation during childhood but also during adulthood. Azoulay et al. (2010) explore peer effects in the field of life sciences. They use quasi-random variation in knowledge flows caused by unexpected deaths of eminent scientists to estimate the effects of collaborations with co-authors.[7] They find a persistent decline in quality-adjusted publication output of co-authors after the death of the eminent scientists.[8] The decline was even more profound in the case of highly cited eminent scientists (Azoulay et al., 2010). Their estimates suggest knowledge spillovers occurring among scientists which are of considerable size.

Though, as Cornelissen et al. (2017) state, these spillovers do not translate into higher wages. They, besides others, specifically look at high-skilled occupations and to what extent peer quality affects workers' wages. On average, an increase in peer quality increases individual wages only to a small extent.[9] They conclude that productivity increases may not be one-to-one converted in wages (Cornelissen et al., 2017). Using wages as a proxy for productivity, thus, likely capture knowledge spillovers incorrectly, whereas innovative output might more accurately capture the peer effects on productivity.

Catalini (2018) further examines the importance of colocation on scientific collaborations. He exploits the exogenous colocation and separation of laboratories at the Paris Jussieu campus, a leading scientific and medical complex in France, due to asbestos removal. The results imply that colocation increases the likelihood of collaborations, which are persistent over future separation. Furthermore, new collaborations tend to work on riskier research as the articles published are

[6]Bell et al. (2019) estimate that the causal effects of the environment explain 75% of the correlation between children's propensity to become inventors and patent rates among adults in their commuting zone.

[7]Unexpected death is defined as an age of 67 years or less at the time of death. Furthermore, scientists needed to be active before death (Azoulay et al., 2010).

[8]Specifically, they estimate a decline of five to eight percent in quality-adjusted publication rates for co-authors of the eminent scientists (Azoulay et al., 2010).

[9]Their results suggest that a ten percent increase in peer quality raises individual wages by about 0.1 percent (Cornelissen et al., 2017).

either at the top or bottom of the citation distribution (Catalini, 2018).

The results imply exposure to other inventors matters and positively impacts innovative activity. The findings likely hold for GitHub users as well. Coding is a knowledge intensive task, and by working together on projects, knowledge spillovers can occur which then may raise the users' productivity. By being geographically close to each other these effects are enhanced, as especially the results by Catalini (2018) show.

2.2.2 Agglomeration Effects and Productivity

Agglomeration effects have been analyzed for a long time. A summary of the work in the field is, besides others, available by Carlino and Kerr (2015). The seminal paper by Jaffe et al. (1993) uses patent citations to measure knowledge spillovers. The researchers find a strong localization of patent citations, which is quite persistent over time. They conclude that, given inventors are more likely citing patents by other inventors that are geographically closer to them, knowledge spillovers are more localized as well. Though, Jaffe et al. (1993) did not further identify city characteristics affecting these spillover effects.

Carlino et al. (2007) build on the work of Jaffe et al. (1993) by analyzing the question of city characteristics determining the extent of productivity effects with an increase in urban density. Especially the employment density, i.e., jobs per square mile, has a positive impact on the number of patents per capita.[10] The competitiveness of the market structure and total employment additionally positively affect patent intensity. The results suggest strong agglomeration effects on the innovative activity of local workers. Living in a denser cluster fosters the knowledge process.

Moretti (2021) investigates how a move of an inventor to a larger cluster affects her number of patents produced and the number of citations received. Both experience an increase after a relocation, which is further confirmed by Instrumental Variable (IV) estimates. Local cluster size is instrumented by cluster size changes that originate elsewhere. This way he tries to tackle a possible bias in the estimates of cluster size due to unobservable productivity shocks.[11] Moretti (2021) concludes that cluster size positively impacts patenting and, based on that, productivity.

The findings show a positive relationship between urban density and productivity, especially for innovative activity. This likely applies for software engineers as well. With a move to a larger cluster, surrounded by a larger number of users in her programming language, the focal user might experience an increase in productivity caused by knowledge spillover effects.

Urban density, next to productivity, also affects other factors such as industrial employment, or firms' location choices (Combes and Gobillon, 2015). Firms should choose to settle where their

[10]Carlino et al. (2007) show that with a doubled employment density, the patent intensity increases by 20 percent.

[11]In detail, Moretti (2021) estimates an elasticity between number of patents produced and cluster size of 0.0662. The contemporaneous effect of a change in cluster size on productivity is 0.0162 and, when instrumented, 0.0307.

expected profit is maximized. As urban density affects productivity and higher productivity results in higher profits, urban density should have an impact on firms' location choices. Research in that area focuses on Foreign Direct Investments (FDI) and firm creations. Market size, measured by local total income or employment in the manufacturing or service sector as well as market access, measured by distance to the main cities in a country, are all found to positively affect firms' location choices (Combes and Gobillon, 2015).

In the context of employment growth, local total employment is used to estimate its effect on employment growth. In general, local market size has a positive effect on industrial employment growth. The effect on employment in the service sector, however, is mixed, depending on the country analyzed (Combes and Gobillon, 2015).

In the following, the focus will be on agglomeration economics affecting productivity and more so innovation.

3 Theoretical Framework

In this section, the theoretical interplay between peer effects, agglomeration effects and innovative activity in the literature is presented to understand the mechanism that may lead to productivity effects on GitHub with an increase in cluster size. To begin with, peer effects and productivity more generally are analyzed. Then, turning to the field of agglomeration economics, the links between peer effects and urban density are explained. To end, the difficulties for estimating agglomeration effects are discussed.

3.1 Peer Effects and Innovation

Invention and innovation are often regarded as similar things. However, as Carlino and Kerr (2015) point out by referring to Schumpeter (1939), invention means the creation of something non-existent to date. Innovation describes the process of putting that service or product up to sell. So the commercial gain of a new product will emerge where it was put in place, i.e., the location of the innovation, which might not necessarily be the same location of the invention.

An innovation has two parts: the creation of a new idea and its commercialization. Both are complementary to each other and necessary for innovative activity resulting in an increase in economic growth (Schumpeter, 1939). If an invention was made at a university in one city, but put in place in another, the immediate welfare effects are rather observed in the latter, where the commercial process took place (Carlino and Kerr, 2015).

These differences are also reflected by patents. Patents inherit by definition a novelty, however not every patent will ever result in a sellable good. Therefore, the rate of patents by location does not necessarily imply higher monetary gains in the case of a higher patent rate (Carlino and Kerr, 2015). In most cases, a patent is never commercialized. For economic growth, though, commercially successful innovative ideas are of interest to analyze (Carlino and Kerr, 2015).

The majority of open source projects contain content for real-world applications, e.g., a project on an email client or a desktop application for SoundCloud, a online music streaming platform (Borges et al., 2016). Commits to these projects, thus, should inherit commercially important input.

Innovations can be distinguished in many dimensions. Either in their form of impact, incremental or radical, or in their form itself, as an improved product or rather an improved process. To find appropriate measures capturing these aspects is arguably difficult. Patents, despite their flaws, embody to a high degree innovative activities (Carlino and Kerr, 2015).

An increase in patents is often seen as an increase in productivity. In times of a rising number of non-routine tasks and learning on the job, peer effects and collaborations undergo an increase in importance as determinants of productivity. Knowledge flows among workers seem to raise their productivity and innovative activity. These knowledge spillover effects occur in collaborations, but not by simply working next to each other (Borjas and Doran, 2015). As expected, knowledge spillovers specifically matter in high-skilled and high-innovative occupations. In low-skilled occupations, it seems, peer effects increase the productivity of workers rather via the channel of social pressure (Cornelissen et al., 2017).[12] Though, as especially innovation tends to cluster to a greater extent, one source of agglomeration economics are knowledge spillovers between individuals (Carlino and Kerr, 2015).

Even though coding itself is a solitary task, collaborations and knowledge spillover effects among software developers matter as well. As in most non-routine tasks, individuals benefit from interactions with others, because, based on them, they can benefit from differences in cognitive frameworks and value sets. These interactions then lead to increases in their own skill set, e.g. by learning a new programming language or new commands in a programming language (Casalnuovo et al., 2015).

For new ideas, which then lead to innovations, especially collaborations matter. The advantages of teamwork for innovative activities are observed by an increasing share of multiple-authors journal articles across all research fields. Patents similarly are filed by a rising number of teams (Wuchty et al., 2007). A large number of factors can be the source of this pattern, varying in importance across research fields. An increase in capital intensity in research might lead the shift towards collaboration in laboratory sciences. Another factor might be the overall increase in researchers and thus, fostering specialization and more diverse teams. With a higher number of researchers,

[12]Social pressure in the context of peer effects describes the feeling of shame or guilt due to lower productivity in comparison to co-workers. Workers might act on that by increasing their productivity (Cornelissen et al., 2017).

the individual researcher is able to focus more on a narrow topic and acquire profound knowledge. Teams of such specialized researchers then cover a broader range of topics and hence, they are more diverse. Moreover, a reduction in communication costs possibly decreases the social network losses which in turn makes collaboration work more efficient (Wuchty et al., 2007).

These findings imply that individual effort and peer performance are complements. An individual only gains from a collaboration by raising her own efforts. The increase in individual effort in collaborations is based on the accumulation of new knowledge as a result of exposure to better peers. The extent of this increase describes the importance of knowledge spillovers as a link between the two (Cornelissen et al., 2017).

Knowledge spillovers in innovative processes depend on the proximity between the peers for which they occur. Proximity can refer to the closeness of intellectual content or geographic distance. Both seem to play a role. Peers that are geographically close to the individual might help her in the workplace, e.g., showing how something works. On the other hand, peers that are close in the intellectual space might foster new output creation within an individual by their ideas (Azoulay et al., 2010). In both cases, peers complement an individual's work via their proximity to her, and thus, increase her productivity, however in two different ways. Both are likely independent of each other, implying a greater increase in productivity as a result of both being present at the same time (Azoulay et al., 2010).

Additionally to the two factors, Borjas and Doran (2015) add the sphere of collaboration networks to the factors. It matters if individuals just work next to each other as colleagues, but might not interact very much or if they have high interactions as a result of collaborations.

These spillover effects via all spheres seem to be the highest if the peer inducing the effects is of high importance. Often, this is measured by the number of citations. A highly cited peer tends to have the highest spillover effects on her co-workers. These individuals might have very innovative ideas and explore fields that receive less attention. They are able to lead the focus of their surroundings to new topics (Azoulay et al., 2010).

3.2 Agglomeration Effects on Productivity

The sphere of geographical proximity is the focus of the literature on agglomeration effects. Agglomeration effects describe the benefits of a high urban density on several aspects as wages, productivity, or incomes. Cities with higher urban density are places of higher labor and production costs. More firms compete for workers, which in turn raises the wages. The increase in production costs is caused by sparse land, as more firms for production as well as individuals for living, demand land. As a result, the production costs in cities are higher than in locations with a lower urban density. Yet, firms are willing to accept these costs due to production benefits that seem to increase with urban density. Agglomeration economics tries to identify the underlying determinants resulting in the advantages of cities (Rosenthal and Strange, 2020).

3.2.1 Causes of Agglomeration Effects

Marshall (1890) states three factors for agglomeration economics, namely, input sharing, labor market pooling, and knowledge spillovers. The latter is described as an unplanned process and requires the highest proximity (Marshall, 1890). It seems, that in the case of knowledge spillovers, communication costs increase to a greater extent with distance. That causes knowledge spillovers to be more localized to foster the unplanned process. New information technologies that offer a reduction in communication costs in more distant interactions are complements to in-person interactions, not substitutes (Rosenthal and Strange, 2020). Theory predicts decreasing marginal returns in productivity gains as a result of agglomeration economics. The gains of an additional skilled worker likely decrease, the higher the number of already present skilled workers is (Combes and Gobillon, 2015).

Agglomeration effects also seem to matter to a varying degree for different types of workers. Rosenthal and Strange (2012) find that the gains of agglomeration are smaller for female entrepreneurs than for male entrepreneurs. Thus, the increase in communication costs with distance differs between workers. This implies heterogeneity in the productivity effects with an increase in cluster size.

Research clusters tend to evolve close to universities as they complement the knowledge creation and transmission. The closer, the better knowledge spillovers can occur. Andersson et al. (2009) suggest that about half of the productivity gains due to agglomeration effects are located within eight kilometers of a newly founded university. Hence, agglomeration effects decay rapidly with distance. A great number of research underlines this hypothesis by providing evidence for knowledge spillovers at the metropolitan level (Moretti, 2021; Rosenthal and Strange, 2020).

This is in line with other findings regarding the heterogeneity in agglomeration effects. In most industries, positive agglomeration effects can be found, except agriculture. The results are as expected, as agriculture relies more on free land (Foster and Stehrer, 2009). Manufacturing on the other hand benefits more from agglomeration effects than the service sector (Melo et al., 2009). This in some aspects contradicts the findings of others about cognitive and social skills being rewarded more in denser cities than motor skills and physical strength (Bacolod et al., 2009). Some suggest that only non-routine occupations gain from agglomeration (Andersson et al., 2014). The manufacturing sector is rather seen as a more routine-intensive sector, whereas the service sector depends more on social skills and non-routine tasks (Autor et al., 2003). The higher benefits of the manufacturing sector due to agglomeration effects may be that the production process is enhanced by closely located suppliers.

The mechanisms resulting in higher productivity with larger city size range from task specialization, worker mobility between firms, labor pooling and training. The hypothesis of task specialization implies that with a larger local labor market, a finer division of labor is possible. Workers may become more specialized and therefore more efficient and productive in their tasks. This process is enhanced by a larger city and, as a result, a larger local labor market (Combes and Gobillon, 2015). Knowledge spillovers might evolve through worker mobility between local firms. A new more productive worker in a firm might increase the productivity in the given firm. Evidence supports this by less on-the-job-training in larger cities (Combes and Gobillon, 2015).

A further mechanism regarding learning as a cause for productivity gains with increases in city size may be that workers at the start of their career move to a bigger city to improve their skills via interactions with more experienced workers. The latter may stay in cities to pass on their knowledge to future generations (Glaeser, 1999).

Next to a better transmission of knowledge, the creation of new knowledge may be enhanced with an increase in city size. A theory by Duranton and Puga (2001) suggests that a larger city may be more diverse. It provides many opportunities for trial-and-error in the product development process and via that, results in more innovations. Therefore, young firms should settle in larger cities and, when being more mature and settled with their products, relocate to smaller towns. In France, this pattern of firm relocation could be found (Duranton and Puga, 2001).

Finally, communication as a means of knowledge spillover seems to also be enhanced in cities. Via communication, knowledge can be transferred between workers, and as suggested, communicative activity is stronger with urban density (Charlot and Duranton, 2004).

It seems for coding, as a skill-intensive and non-routine task, positive agglomeration effects on productivity should take place. Furthermore, GitHub as a coding platform with social characteristics likely further fosters these effects.

3.2.2 Endogeneity Concerns in Estimating Agglomeration Effects

There are several difficulties when estimating agglomeration effects on productivity. One is regarding the quality and quantity of labor. High-skilled workers may self-select themselves to work and live in a larger city to a greater extent than low-skilled workers. This may be because they value the amenities a larger city offers more, or because, for historical reasons, high-skilled workers tend to reside more in cities and pass on their skills to future generations. This could, next to an endogeneity concern, also be a case of reverse causality. As a result of more amenities and productivity gains in a larger city, urban density increases and a higher presence of high-skilled workers in cities may not be the result of agglomeration effects (Combes et al., 2010). Moreover, more productive workers may choose a larger city to have a higher benefit of the productivity gains of a city. In that case, individual ability and urban density would be correlated (Combes et al., 2010). Not taking these factors into account would lead to a bias in any estimation of the causal effects of urban density on productivity. A way to overcome this bias may be by using individual panel data and introducing individual fixed effects. In that case any time-invariant individual characteristics affecting their productivity is controlled for and, hence, the bias reduced (Combes and Gobillon, 2015).

Another concern are local productivity shocks that simultaneously affect urban density and productivity. For instance, a city might invest a lot in research and via that attract new inventors, increasing urban density and next to that increase productivity. As a result, this leads to a bias in the estimate (Combes and Gobillon, 2015). Different ways to address this concern were introduced in the literature. For instance, by introducing cluster or city fixed effects as well as interaction effects of city and time, these would capture time-invariant and time-variant cluster characteristics introducing a possible bias in the estimate (Combes and Gobillon, 2015). They would not, however, resolve a reverse causality issue, meaning an increase in productivity translating into more amenities in a city. This in turn would attract more high-skilled worker. Historical and geographical instrumental variable approaches were introduced to mitigate these biases. For instance, long lagged values of population or density are considered relevant as they likely affect urban density nowadays, however assumed to be exogenous to unobserved local productivity shocks. Other instruments consider the subsoil of a location (Combes et al., 2010).[13]

Alternatively, shift-share instruments are implemented by using exogenous aggregate shifts to predict local changes (Farhauer and Kröll, 2009). If, e.g., the employment in other software firms in other cities increases, the local employment in the focal software firm may also increase. The geographical variation in software firms is used to predict changes in local employment in a software firm. In that case, local productivity shocks likely do not affect the instrument, i.e., employment changes in other cities.[14]

The dynamics of agglomeration effects are a minor cause of bias. In most specifications, they are assumed to be contemporaneous. Although, it could be the case that density effects have not only an impact on productivity in the same period but result in an ongoing raise in productivity. Furthermore, agglomeration effects of one city may also affect neighbouring cities. Yet, as agglomeration effects seem to decrease rapidly with distance, this seems a minor concern (Combes et al., 2010).

[13]The considerations in that case are such, that soil composition or depth to rock were important determinants of agriculture in the past. The developments from agriculture to manufacturing and service occurred in places where people already settled. These instruments are possibly good predictors of urban density nowadays, however, are unlikely correlated to productivity gains in present times in a city (Combes and Gobillon, 2015).

[14]Moreover, natural experiments or Generalized Method of Moments (GMM) estimation are further alternatives used in the literature to mitigate the concern of biases in the estimates (Combes and Gobillon, 2015).

4 Research Design

Now turning to the empirical framework, first, the platform GitHub is explained in more detail. Next, the steps for the data preparation as well as the data limitations are discussed. Then, the data is described and the estimation strategy is presented.

4.1 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 2007 and since then experienced an increase in popularity across software developers (Fackler and Laurentsyeva, 2020). Its attractiveness lies in the easy usage and, in its basic version, no costs. 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 repository to which code can be *pushed*, i.e., uploaded. The platform supports every programming language. Each repository has one owner. After invitation, other users can become project members. They can modify the repository's content and approve or disregard submitted contributions by others (Lima et al., 2014). A repository can also be *forked*, i.e., copied, and in this case independently worked on. A *pull request* represents the sum of code changes (*commits*) a user sends to the repository in a session. Collaborators can review and *merge* it into the repository or discard it (Lima et al., 2014). Regarding the social features of GitHub, it is possible to *follow* other users and then be notified about their actions. Another trait is to *star* a repository. This way, it is bookmarked and can be found more easily later in time. The number of stars or forks per project is seen as a measure of a project's popularity among users (Lima et al., 2014).

When registering, users are able to provide their real name, location and other biographical information (Fackler and Laurentsyeva, 2020). Each repository 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 repository are observable by everyone (Fackler and Laurentsyeva, 2020). Furthermore, on every user page their actions are shown and everyone can see to which projects a user contributes as well as the timing of it (Laurentsyeva, 2019).

The social features of the platform allow users to develop impressions of other users' social and technical skills and behavior (Casalnuovo et al., 2015). For instance, Tsay et al. (2014) find that users next to forming opinions about the abilities of others also use these features to control their own online standing.

The motivation for contributions on OSS platforms was analyzed in the literature and the findings suggest heterogeneity across individuals. The driving motives spread 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, 2008; Hergueux and Jacquemet, 2015)

Positive productivity effects of co-workers on individuals seem to be mitigated in an environment of fixed wages. The source may be a free-rider problem. If a more productive co-worker enters the team, other individuals might decrease their effort because of the more productive worker taking up a greater share of the work (Herbst and Mas, 2015). The motivation on GitHub is often intrinsic and, hence, users likely won't fall into the trap of a free-rider problem, but rather knowledge spillovers lead to an increase in productivity.

The social characteristics of the platform affecting collaborations among software engineers was studied as well. Users are more likely to join the projects of users they have social connections with (Casalnuovo et al., 2015). In these cases, productivity, measured by a user's number of commits, is enhanced at the start of the collaborations as well as in the long run. The authors also find that tighter social connections lead to lower productivity in the beginning, but larger increases in the long-term (Casalnuovo et al., 2015). One reason might be, that in the beginning of a project, where the new member has to understand the project structure, more prior links lead to more communication and less output. After this initial phase, a user then might be able to focus on specific parts that raise output and productivity (Casalnuovo et al., 2015). These developments depend on the one hand on the programming language. If a user is more familiar with it, productivity increases are higher. On the other hand, it depends on the level of social connections with other project members. If they are stronger, the productivity increase is further enhanced (Casalnuovo et al., 2015).

A feedback and recognition system as well as a community infrastructure increase the contribution's quality (Wright et al., 2020). On GitHub, users can add comments to their commits, and that way send feedback to the other project members. Timely feedback was found as an important factor for innovation incentives (Manso, 2011).[15]

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. Similar to the findings of Azoulay et al. (2010) about greater productivity increases due to collaborations with eminent scientists, individuals also value more collaborations with more established users. They seem to raise their own motivation and their impression of possible project success (Casalnuovo et al., 2015).

Acquiring knowledge through work with others on OSS platforms was also found to positively impact entrepreneurship. Through collaborations, users make links which then may lead to starting entrepreneurial businesses (Wright et al., 2020).

Thus, agglomeration effects may also affect the productivity of software engineers. As a result of a move to a larger cluster, a GitHub user can form more social connections with other users in

[15]Manso (2011) shows that timely feedback on performance especially matters for exploration. It gives the individual information to improve future work.

the cluster. Based on these connections, the user will contribute to a greater number of projects with a higher number of commits and a productivity increase is observed. Especially in GitHub projects with a small number of project members, users tend to be geographically close to each other (Casalnuovo et al., 2015). This would be in line with the mechanism found by Jaffe et al. (1993). Instead of local inventors citing patents by other local inventors, now users commit to projects by other local users.

4.2 Data

In this section the generation of the data set is described. Furthermore, the limitations of the data are discussed and their possible effects on the results and the results' interpretation.

4.2.1 Commit Data

The full data used for the analysis is a combined version of several snapshots from GitHub Torrent (GHTorrent) (Gousios, 2013). GHTorrent creates snapshots of the activities on GitHub, e.g. user registration, projects and commits, and makes it accessible to everyone in a relational database. The commits recorded are only commits to public repositories.

The snapshots included in the 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). The activity stream of commits as well as data on the corresponding projects are taken from the snapshot 202103. The commits queried were limited to users that have a US or Canadian location stated in the respective snapshot.

Every user has a unique user id. Commits are matched via the author id, not the committer id, to the user id. The scope of the analysis is to observe productivity changes based on changes in cluster size, i.e., if a user produces more (new) output measured by more commits. This is more likely captured by more written commits than uploaded commits. Matching the commits by committer id might rather capture a higher activity on GitHub more generally and a link to higher productivity is less clear. The user might create a lot of pull requests with other users' written content. Hence, matching commits by author id to users creates a better image of the user's knowledge output. Some users have several GitHub accounts (Casalnuovo et al., 2015). Unfortunately, we cannot account for these user aliases and might underestimate spillover effects as a result of less commits per user than she actually contributes.

The commit data contains all commits a user has ever generated after account creation until the date of the snapshot. Commits are matched via a project id to the project table to obtain information on the project's programming language. This way the commit's programming language is identified.[16] The programming language can be understood in a broader way and includes frameworks or databases as SQL. For the analysis, only projects with a stated programming language are considered. Further information about the project, e.g. project creation date, number of watchers or number of forks are added via the project and watcher table.

In total, there are 404 stated programming languages[17] in the commit dataset. However, 18 programming languages cover 90 percent of all commits. These are C, C#, C++, CSS, Go, HTML, Java, JavaScript, Jupyter Notebook, Objective-C, PHP, Python, R, Ruby, Rust, Shell, Swift, and TypeScript. To aggregate programming languages, CSS and HTML are combined as they are commonly used together. The programming languages are further assigned a technology class 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 by developers jointly.[18] We focus on the stated 17 programming languages and match them to the technology clusters which, in the end, leads to five technology fields. Cluster one contains JavaScript, CSS, PHP, C# and TypeScript. Cluster two Python, Shell, Go, Jupyter Notebook and R, cluster three Ruby, cluster four Java, Objective-C and Swift, and cluster five C++, C and Rust.[19]

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.[20]

If a user commits in a programming language other than the ones included in the analysis, the user is recorded with zero commits for the technologies used beforehand, given they are included in the sample. For example, if a user commits in technologies one and two in time interval one, in FORTRAN, a programming language which is not included, and thus not assigned a technology to, in time interval two and in technology one in time interval three, for time interval two zero commits are filled for the technologies one and two, the technologies used in the previous time interval. If a user only commits in a technology not included in the sample, she is excluded.

To remove inactive accounts, the data was further limited. It was filtered for only users that commit in at least two time intervals or, if the account was created in the last time interval and the user committed in that time interval, those users are included as well.

[16]In a project, files in several programming languages can exist. GHTorrent defines the project's programming language by the programming language that makes up the largest number of byte counts in the project.

[17]We use the term "programming language" in a broad sense (for example, including markup languages) to distinguish them from natural languages.

[18]See Figure 7 for a visualization of the correlated technology clusters.

[19]Cluster 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.

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

4.2.2 User Data

Regarding the user accounts and their location, a snapshot contains only the currently stated location. To observe user location changes, the snapshots of the user accounts from 201509 until 202103 were combined. Accounts can be marked as fake, i.e., that a user appears only as a committer or author of a commit, but does not have own projects or creates other events as push or pull requests. Those users are included in the sample as we are only interested in commit events and not in other events such as project creations.[21]

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.

The user table contains a variable *location* which contains the self-stated user location. However, in about 90 percent of cases, i.e. 210,705,552 observations, no location information is available at all. In these cases we use the non-empty location of the snapshot before if available and otherwise of the next snapshot.

The non-empty location information is used to geocode the user. In practice, 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 on the cities. For US and Canadian cities, further, more comprehensive data sets provided for free by simplemaps was used (Simplemaps, 2021).

Users are further matched 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 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 404,651 users with always US or Canadian locations and matched to economic areas, with 3,020,587 user snapshot observations.

4.2.3 Full data

The combined commits and user data results in 10,785,249 observations. The full data set is used to calculate the cluster size. For the regression, only users that are observed in all snapshots, i.e. geocoded and with non-zero commits in all time intervals, are used. These are 2,238,606 observations and 18,377 unique users.

In section 4.3 both, the full data and the regression data are described in detail. They are

[21] Accounts marked as fake are about 22.8 percent of all user observations.

very similar regarding the distribution of commits per programming language and per technology. 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. This suggests that those users are more likely to experience productivity effects in larger clusters and that we are able to observe this by their high, and possibly even higher activity on GitHub. 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 result in clusters actually too small. 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 length of time intervals with non-zero commits per user. In this specification, the elasticity becomes less significant.[22]

4.2.4 Clusters

The clusters are constructed as technology × 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 speaking, cluster size is calculated as:

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

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 per 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 × city and 2 × city in time interval one.

4.2.5 Data Limitations

(1) The first main limitation of the data are user locations. Location changes are only observed if a user decides to change the location in her account. Hence, it depends on how thoroughly and regularly one cultivates one profile. Less active users might also put less emphasis on updating

[22]See section 5.6 for the discussion of the results.

their account on a regular basis. However, these users might also benefit less from denser clusters or at least it would be more difficult to tell given that only a smaller number of commits is observed for them and the location changes may be delayed. On the other hand, if they become more productive, i.e., commit more after a move before updating their location, it would lead to a downward bias in our estimates. The relevant cluster size would be measured incorrectly, as a user is assigned to a city she actually is not currently living in.

A large number of users in our full data, that is, 14,844 users, have no location information at all in nine snapshots and are assigned their earlier or later location. In these cases, we assume no location changes and the measurement error is further magnified. This measurement error in the independent variable, cluster size, should lead to a bias towards zero, also called attenuation bias (Pischke, 2007).

On the other hand, users included in the regression are users, that are active in all time intervals and, hence, use GitHub on a regular basis, which makes them more likely to update their accounts. The measurement error in the independent variable, namely, cluster size, possibly occurs for less active users that are not part of the analysis.

(2) A second limitation stems from the fact that the data contains only commits to public projects. If, for example, users in larger clusters tended to contribute more to private projects, we might observe a decrease in public commits. We would be unable to tell this shift apart from other effects causing a decrease in public contributions. Another possible reason might be that a user spends less time committing as a result of a new job, and hence, an actual decrease occurs and not a shift. Yet, we cannot distinguish the two with our data. The reasons to contribute to OSS projects are very broad and there are different kinds of public projects. Some are leisure projects, users work on next to their main job. Other projects are work projects, either for users' work or as their main job. These work projects can also be a way for a user to show her skills and, via labor market signalling, attract job offers. Then again a decrease in commits might be because the worker has a new job and is less active on GitHub and cannot be interpreted as a decrease in productivity.

(3) The third limitation is the content of commits. Commits are any change of code. This can be correcting a typo, but also making major changes in the code. A commit could also be uploading files to a project to save them on GitHub. In that case, a user might commit several times a day. Therefore, the extent of knowledge creation might vary significantly across commits. However, we do not analyze the content of the commit and might compare a typo correcting commit with a major code changing commit. To reduce this possible concern, in a further analysis we restrict the sample to users with commits to only projects with at least one star. The number of stars or

the number of forks per project are quality indicators for a project. A commit to a project with more stars or forks is more likely to be of higher quality as it is more difficult to commit to such a project. Nevertheless, we still cannot completely rule out the case of minor code correction in these commits neither.[23]

A possible solution for future work could be to further examine the content of the commit. Casalnuovo et al. (2015) take the number of files touched with commits or number of changed code lines, either added or deleted, as productivity measures of GitHub users.[24]

4.3 Descriptives

In this section, we are going to describe the full data, i.e., the data of all users on which cluster calculation is based on. This data set will be compared to the regression data, in which only users with commits in all time intervals are included.

4.3.1 Commits, Users and Projects

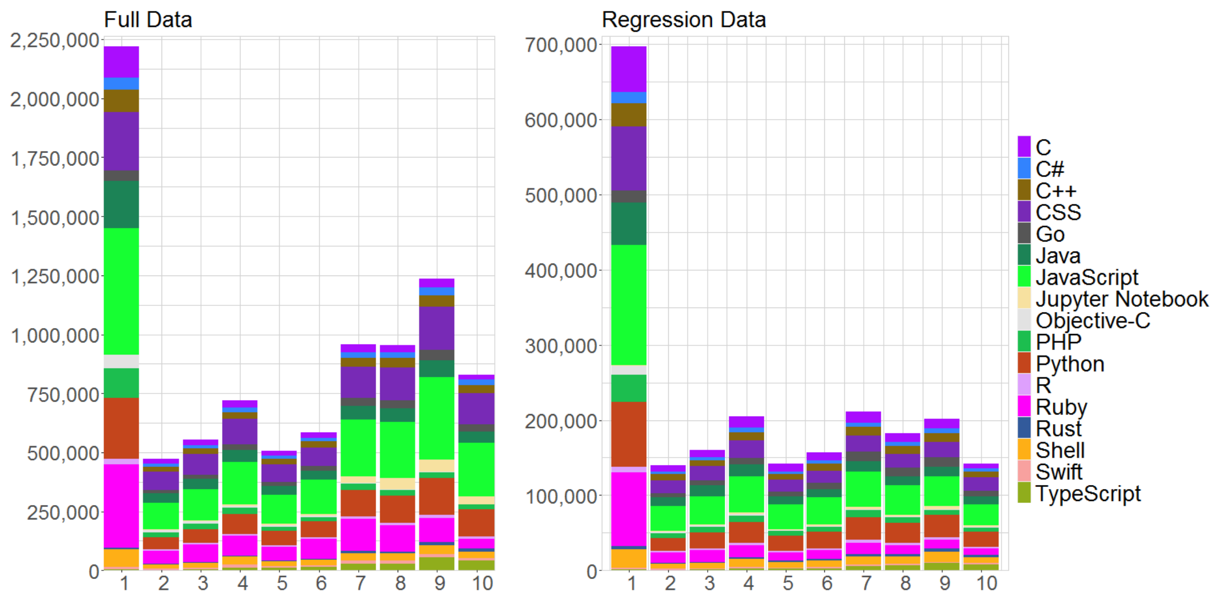
The left graph of Figure 1 shows the number of observations per time interval and per programming language in the full data.

Part of the variation 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 an account was created 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, time fixed effects are included to take this variation of commits into account. The distribution of observations in the regression data, shown on the right of Figure 1 is similar to the full data, but the level of observations, as expected, is a lot smaller. The number of observations per time interval is cut in half or decreased even more when restricting the sample to only users observed in all time intervals.

[23]Additionally, a commit might be assigned to a different programming language than the one it is actually written in. The commit's programming language is determined by the project the commit is to, but a project can contain files in several programming languages. The 'main' programming language of a project is based on the largest number of byte counts. However, this should cause a minor measurement error, as the majority of commits to a project are assigned to the correct programming language.

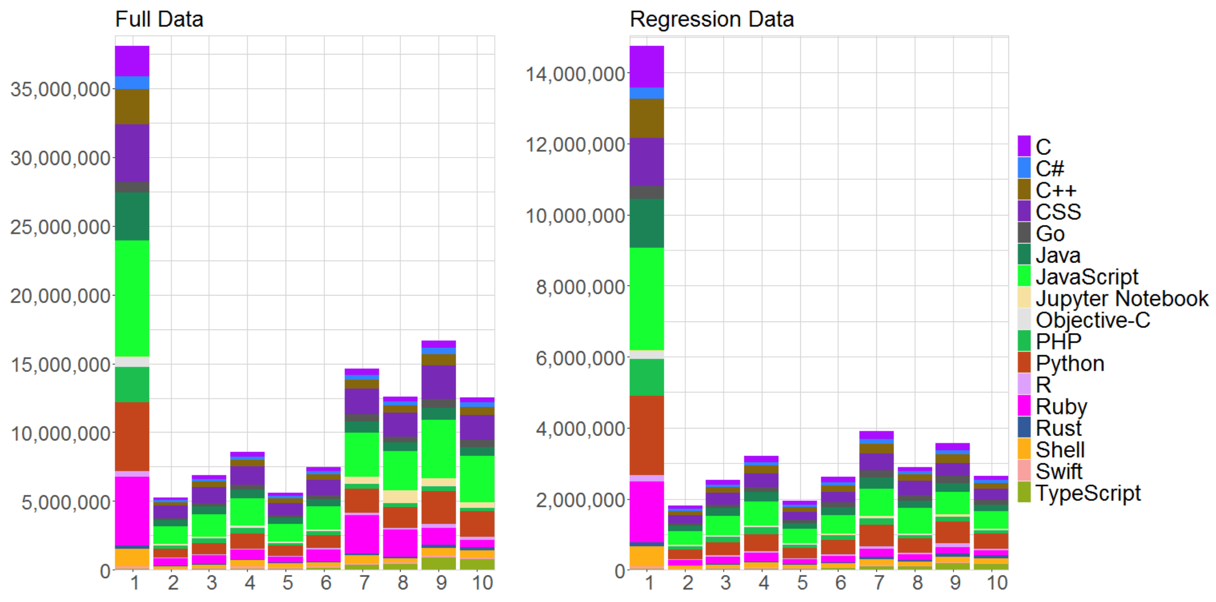
[24]As Casalnuovo et al. (2015) note, using the number of changed code lines might come with noise. Users could copy code from other files and add this to a file or merge codes. This leads to imprecisely measuring productivity. However, they compare their results based on line changes and file changes and find that the results are consistent across measures.

Figure 1: Number of Observation per Snapshot and per Programming Language



In both figures, there is noticeable an overall increase in users, projects and commits over time, similarly shown in Figures 2 and 5. The distribution of programming languages in the observations is comparable to the distribution of the programming languages in the number of commits as shown in Figure 2. It plots the sum of commits per programming language for all 17 programming languages.

Figure 2: Sum of Commits per Programming Language per Snapshot



To take into account differences in the number of commits per programming language, programming language fixed effects as well as programming language $\times$ time fixed effects are introduced in the regression. The latter controls for time trends in the usage of programming languages.

Comparing the regression data with the full data in Figure 2, the distribution of commits per programming language is also very similar, but levels are lower.

Table 18 shows the share of commits per programming language for the full data and Table 19 for the regression data. The programming language effects in the regression control for the differing popularity of programming languages.

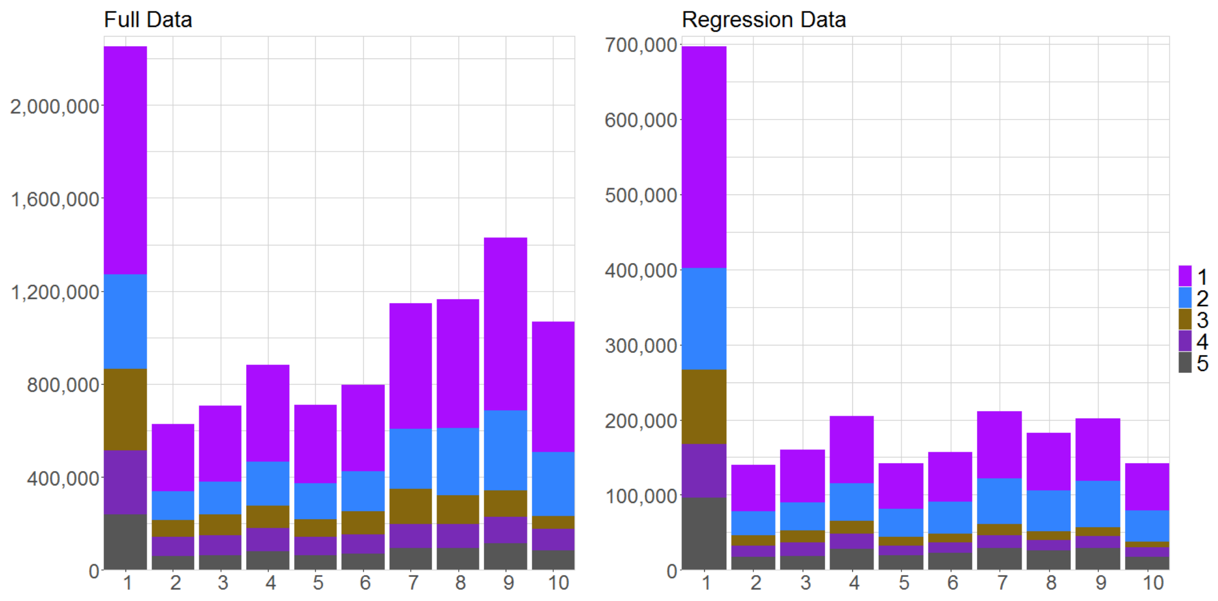
It shows the minimum, median, mean, and maximum total number of commits per user for the 17 programming languages in the full data. JavaScript has the highest median total number of commits per user with 25, followed by CSS and Ruby each with 18 commits per user.

The total number of commits per programming language per user is very broad. The maximum total number of commits per user is 364,996 in JavaScript, followed by Shell with 281,997. An explanation for these high numbers for users might be that they are very active on the platform because they use GitHub for their work.

Regarding the regression data in Table 19, first of all, the median increases for all programming languages. The fact that the maximum number of total commits for most programming languages changes only slightly shows, that the regression data includes more active users.

The number of observations and sum of commits per snapshot, each per technology, are again very similar in full and regression data, shown in 3 and 4.

Figure 3: Number of Observation per Snapshot and per Technology

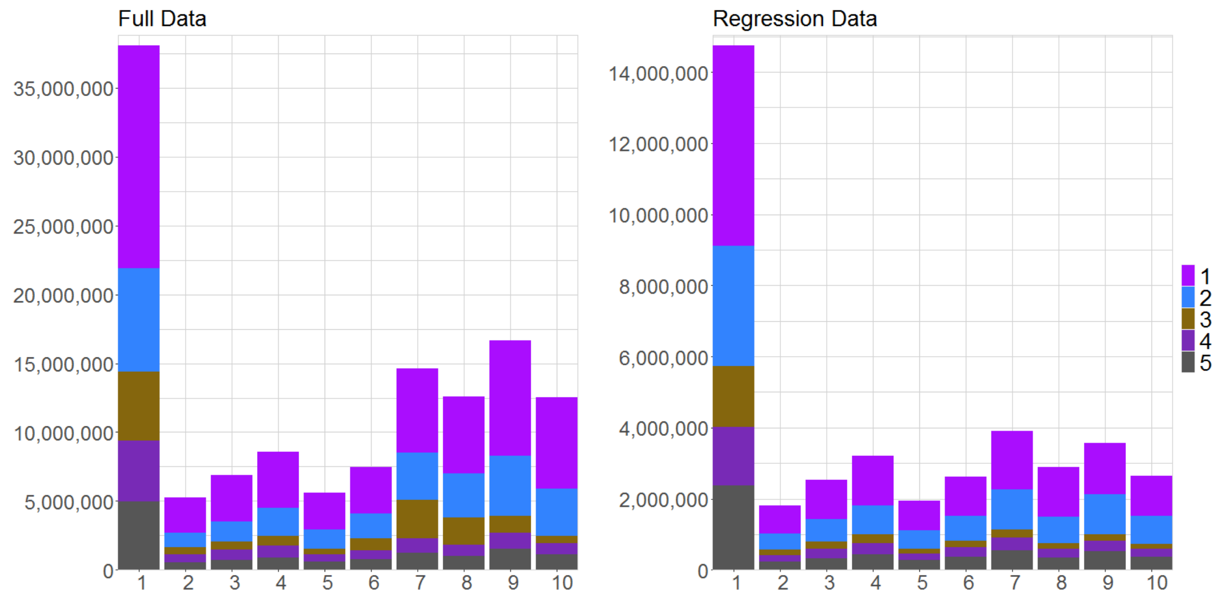


Clusters one and two contain a higher number of observation and sum of commits as they are comprised by a larger number of programming languages.[25] Remarkably, cluster three, contain-

[25]Cluster one contains JavaScript, CSS, PHP, C# and TypeScript. Python, Shell, Go, Jupyter Notebook and R

ing only Ruby, is very similar in distribution to cluster four, which contains three programming languages.[26]

Figure 4: Sum of Commits per Technology per Snapshot



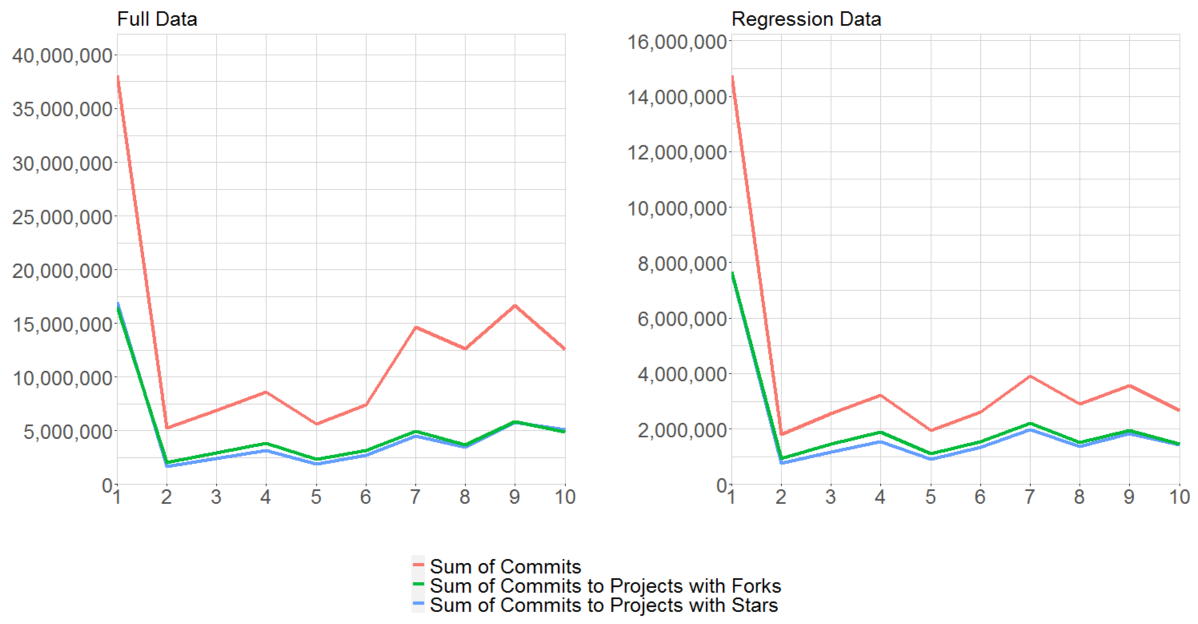
This is also noticeable by median number of total commits per technology shown in Figure 20 and 21. Technology one makes up about half of all commits (46% in the full data, and 41% in the regression data), followed by technology two (23% in the full data, 26% in the regression data), three (11% in the full data and 8% in the regression data), four (9% in the full data and 10% in the regression data) and five (10% in the full data and 15% in the regression data). Hence, even though clusters vary in their number of programming languages, their distribution in the data is relatively similar. Only cluster one makes up a very large share. However, it also contains JavaScript, the most used programming language, which by itself already makes up about 23% of all commits.

Therefore, next to programming language and programming language × time fixed effects, we include also technology fixed effects to account for differing general popularity of technologies.

are in cluster two.

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

Figure 5: Sum of Commits per Snapshot



If one looks at the sum of commits shown in Figure 5, it is increasing over time. The difference between the total sum of commits and the sum of commits to projects with at least one star in Figure 5 is a result of the great number of projects with zero stars. The lines for the sum of commits to projects with stars or forks show only the commits to non-forked projects with stars or forks.[27]

The number of stars and forks are an indicator for the quality of a project (Laurentsyeva, 2019). Commits to projects with zero stars or zero forks might be the user's own project in which she possibly only saves files. These commits might not necessarily be an indicator for productivity. The distribution of stars, shown in Table 22, is highly skewed towards zero. For non-forked projects with at least one star, the median number of stars is four while the highest value of stars per project is 259,118. 75 percent of all non-forked projects with at least one star have between 24 and one star. The distribution of the number of forks per project is very similar. The median number of forks for non-forked projects with at least one fork is two and about 75 percent of those projects have seven forks. The maximum number of forks is 145,997.

This suggests, there are a lot of 'small' projects regarding the number of forks and stars. To commit to a project with a great number of stars or forks demands a higher quality commit. Though, this might take more time and the absolute number of commits might be smaller than the number of commits to a project with less stars or forks. To account for this quality concern of commits, the baseline model is used with a sample restricted to users that commit in all time intervals to only projects with at least 100 stars and also only those commits are considered. Hence

[27]Non-forked projects are the projects at the root of forks, one could say the original project. They are not forked from another project.

the sample is restricted to potentially more high-quality users with more high-quality content.

General popularity of projects, which the number of stars or number of forks are proxies for, likely affects the total number of commits to a project. Project fixed effects included in the regression account for this aspect.

Another distinction can be made between contributions to a user's own project and to other's projects. About 73% of projects in the full data and, similarly, 50% in the regression data are not the user's own project. Regarding the commits, 62% in the full data and 56% of commits in the regression data go to projects of others.[28] This suggests that the projects in the sample are not necessarily 'toy' projects or projects a user might only use for saving files.[29] The project age, i.e. the time in years it was created until the latest snapshot date, supports this assumption. In the full data, projects are about four years old, in the regression sample five years old. As OSS projects go through different stages in their development process, an older project hints at being more mature.[30]

OSS projects are often managed by firms (Riehle, 2012), which may suggests that users in our sample are full-time OSS contributors. The share of commits by project made during business hours, i.e. Monday through Friday from 6am to 20pm, supports this hypothesis. About two third, 59% in the full data and 62% in the regression data, are made during business hours. On the other hand, the share of commits made at the weekend is only 21% in the full data and 19% in the regression data. So, it seems the projects tend to mainly be work projects.

In the full data, the total number of commits per user, considering all programming languages, shown in Table 22, ranges from one to 388,287 commits. Again, it might be that the user with a total number of 388,287 commits, uses GitHub for work, although we cannot confirm this by the data at hand.[31]

The most active user in the regression data has 235,640 total commits, shown in Table 23. Concerning the average total number of commits in the full data, it is 317.08 compared to 2,171.40 in the regression data. The median is 78 total commits per user in the full data and 1,089 in the regression data. This strong increase in the median is partly a result of the data definition. This

[28]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.

[29]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.

[30]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.

[31]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.

shows again that the more active users remain in the regression sample. User fixed effects in the regression model control for the differing activity levels across users.

The number of forks and stars of the projects included is higher in the regression data. While 75 percent of projects with at least one star in the full data have between one and 24 stars, it is between one and 58 in the regression data. Similar occurs for the number of forks, given a project was at least once forked. The number of forks for 75 percent of non-forked projects with at least one fork in the full data ranges from one to seven in comparison to one and 14 in the regression data. Next to the more active users remaining in the regression data, also projects with higher quality stay in the data. The quality of more active users' commits is higher, as they, with higher activity level, possibly have more experience on GitHub and the programming languages. Then, they are able to commit to projects that demand higher quality of the commits.

Out of the 404,651 users in the full data, 46,517 users moved in sum 52,300 times.[32] In the regression data, 4,331 of the 18,377 users moved a total of 5,196 times.[33] This suggests sufficient variation in cluster size.

On average, users use a total of 4.25 (median: 4) programming languages and similarly, on average per time interval 2.12 (median: 2) programming languages in the full data. In the regression data, again users are active in more programming languages. The average number of programming languages used in total is almost twice as high as in the regression data with 6.53 (median: 6). Per time interval, the average number of programming languages is slightly higher with 3.25 (median: 3) programming languages

Regarding technologies, users are on average active in 2.24 (median: 2) technologies in the full data and per snapshot on average in 1.68 (median: 1) technologies in the full data. In the regression data, users are active not only in more programming languages but also technologies. Users commit on average in 3.45 (median: 4) technologies and per snapshot in 2.26 (median: 2) technologies.

The overall higher activity of users in the regression data suggests that they may experience the largest spillover effects. They likely interact the most with other programmers and benefit the most from exposure to different skill sets. Furthermore, shifts in productivity should also more observable for these users.

[32]41,159 users moved once, 4,961 users moved twice, 369 users moved three times and 28 users moved four times. Moves occurred in the second time interval 8,295 times, 2,350 times in the third time interval, 6,120 times in the fourth time interval, 148 times in the fifth time interval, 34 times in the sixth time interval, 13,302 times in the seventh time interval, 278 times in the eighth time interval, 55 times in the ninth time interval and 21,718 times in the tenth time interval.

[33]3,548 users moved once, 710 users moved twice, 64 users moved three times and 9 users moved four times. Moves occurred in the second time interval 1,395 times, 242 times in the third time interval, 732 times in the fourth time interval, 1,382 times in the seventh time interval and 1,445 times in the tenth time interval.

4.3.2 Clusters

Now we turn to describing the clusters, i.e., changes in cluster size over time. Clusters are calculated on the basis of the full data, and hence, the cluster sizes in the full data and the regression data are the same.

Table 24 shows the largest clusters for the five technologies as of the 202103 snapshot and Table 25 shows the distribution of cluster size per technology.

For all technologies, San Jose-San Francisco-Oakland, New York-Newark-Bridgeport and Seattle-Tacoma-Olympia are the top three largest clusters. Especially for San Jose-San Francisco-Oakland, between 12 to 16 percent of all users in the respective technology in the snapshot 202103 stem from this area.

In the case of technology one, San Jose-San Francisco-Oakland makes up about 12.55 percent of all users in that technology, followed with a profound gap by New York-Newark-Bridgeport with 9.89 percent and next Seattle-Tacoma-Olympia with 6.46 percent. The top ten cities cover 61 percent of all users in technology one in the latest time interval. The ratio between the largest cluster and the median cluster is 256.2. This means about 256 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 18.25. This shows that cluster size decreases rapidly moving away from the largest cluster.

For technology two, San Jose-San Francisco-Oakland is again by far the largest cluster with 15.37 percent of all users stemming from this area. The ratio between largest cluster and the median cluster is 365.93, i.e. 366 times more users stem from San Jose-San Francisco-Oakland than from the median cluster, with 0.04 percent. A lot smaller with 20.62 is the ratio between the 90th percentile and the median cluster. The difference between largest cluster and 90th percentile cluster is even more extreme for technology two than for technology one.

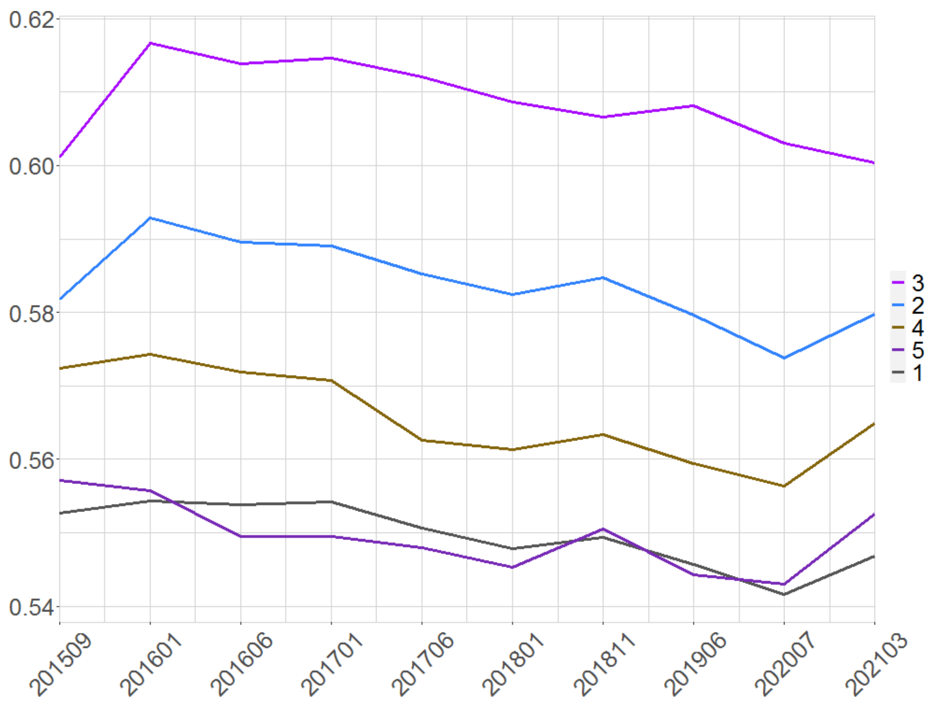
In the case of technology three there is a smaller difference between the largest cluster San Jose-San Francisco-Oakland and New York-Newark-Bridgeport, second largest cluster, with almost 15.55 percent to 12.71 percent. The ratio between the largest and median cluster is 272.74. This means that 273 times more users stem from San Jose-San Francisco-Oakland than the median cluster with 0.06 percent.

For technology four, again the largest cluster is San Jose-San Francisco-Oakland with a cluster size of 16.15 percent and New York-Newark-Bridgeport with 9.11 percent is the second largest cluster. Thus, the distance in cluster size between the largest and second largest is similarly pronounced as for technology one. The ratio between largest cluster and median is 351.02 and 24 between the 90th percentile and median. Hence, again cluster size decreases quite rapidly.

Technology five is very similar to technology three regarding cluster sizes and ratio. The largest cluster is again in San Jose-San Francisco-Oakland with 16 percent followed by New York-Newark-

Bridgeport with 7.66. The ratio between largest cluster and median is 285.73 and 18.73 between 90th percentile and median.

Figure 6: Share of Top 10 Cities for All Technologies



As noticeable in Figure 6, users in the programming languages 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 60 percent of all users in that technology stemming from only ten cities and technology two with also 60 percent in the tenth time interval, the clustering seems to be profound. Though, for most technologies the share over time decreases.[34]

4.4 Estimation Strategy

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

$$ln(y_{ijflct}) = \alpha\ ln(S_{-ifct}) + d_{cf} + d_{cl} + d_{lt} + d_{ct} + d_c + d_f + d_t + d_l + d_i + d_j + \mu_{ijflct} \qquad (1)$$

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

[34]The overall decrease in clustering is in line with findings by Wachs et al. (2022). The authors show that in recent years the strong clustering of OSS activity in the U.S. decreases and is more equally distributed around the globe.

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_c are city effects, controlling for time-invariant city characteristics; d_f are technology effects, accounting for time-invariant technology characteristics; d_t are time fixed effects, capturing snapshot specific characteristics; d_l are programming language effects, accounting for time-invariant programming language characteristics; 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. The error terms for users in a city active in a technology are likely to be correlated with each other, hence clustering should be done on this level. Additionally, treatment, i.e., the change of cluster size, also occurs on the technology $\times$ city level. Variation in cluster size stems from users moving and by users starting and stopping to commit in a technology.

A positive α would imply a positive elasticity between cluster size and productivity. An increase in cluster size, either by more local users committing in the technology or by moving to a larger cluster, is positively related with a user's productivity, i.e. a higher number of commits. This would hint at spillover effects raising the focal user's productivity. The more users are positively associated with the focal user's activity. A negative α would imply that a user commits less to public projects with an increase in cluster size. This would not necessarily imply a decrease in productivity. It could be the case that the user commits more to private projects or that the decrease in commits is due to a new job after moving.[35] In both examples, a negative α does not necessarily imply a decrease in productivity or the nonexistence of productivity spillovers. Hence, interpretations of the results have to be done with caution.

Clusters are defined as the number of users in a city relative to all users in a technology and, thus, the effect of cluster density on productivity is estimated. By including city $\times$ time fixed effects, we take into account changes in city size, and hence the elasticity estimates are similar to holding constant city area (Moretti, 2021).

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 also 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.

[35] About half of all GitHub users work on private projects (Kalliamvakou et al., 2014).

A possible concern in our case could be, that users move in expectation to be more active to a larger cluster. 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 (Moretti, 2021).

The latter concern regarding OLS identification of α are unobserved productivity shocks at the individual or local level. GitHub might foster a technology in a city by promoting local projects in that technology. Users may 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.

Via an instrumental variable approach, in line with 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 and not an outcome of the dependant variable, i.e., the productivity of a user in a local cluster (Combes and Gobillon, 2015). Changes in local GitHub projects originating elsewhere arguably meet these conditions. If a project experiences a higher number of users committing to elsewhere, the number of users committing to the project likely also increases in the local city. These increases somewhere else possibly are good determinants of changes in the local number of users, and via that, the local cluster size. Yet, these expansions elsewhere unlikely are 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, an exogenous and relevant instrument for changes in local cluster size is obtained.

In practice, we implement a variant of a shift-share approach by considering 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 $\Delta N_{jf(-c)t} = N_{jf(-c)t}$ - $N_{jf(-c)(t-1)}$.

More formally speaking 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 nationwide 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.[36]

Identification stems from changes in the number of users originating from other cities committing to local projects besides the focal user's projects. The number of users committing to the focal user's project likely increases if the number of users stemming elsewhere committing to other projects in her city increases.

For example, users from San Francisco and New York commit to a project x. If more users in New 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.[37]

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

[36]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 nationwide change in users in a technology.

[37]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, but necessarily not 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.

other cities than the user's city.

GitHub is a very attractive place 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 commits from different cities should be available to apply a variation of a shift-share instrument.

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):

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

This way, the contemporaneous effect of cluster size on productivity is estimated.

5 Results

In this section, the baseline estimates of equation (1) are presented. To account for possible endogeneity concerns an event study design is carried out. Additionally, a heterogeneity and mechanism analysis as well as robustness checks to test the validity of the results are conducted and shown.

5.1 Baseline Estimates

Table 1 shows the estimates for the OLS regression of equation (1). For this regression, only users which 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 0.1052 (0.1206). After adding city $\times$ technology fixed effects in column two, taking into account time-invariant city-technology characteristics, the coefficient for log size becomes 0.2553 (0.1407) and is now statistically significant at the ten percent level. The coefficient in column two becomes more than twice as high than in column one. This suggests, that in larger clusters, cities have features that lead to users being less active. 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.[38]

The estimates stay significant at the ten percent level after adding dummies for the interaction of city $\times$ language effects in column three. The elasticity increases only slightly from 0.2553

[38]This is also supported by excluding San Jose-San Francisco-Oakland, the largest city for all technologies in the last snapshot, from the sample. The final elasticity becomes even larger with 0.2980 (0.1113) in comparison to 0.2402 (0.1134), both significant at the five percent level.

(0.1407) in column two to 0.2524 (0.1429) in column three.

Table 1: Baseline Estimates

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1052	0.2553*	0.2524*	0.1887**	0.2402**
	(0.1206)	(0.1407)	(0.1429)	(0.0766)	(0.1134)
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
City x Technology		Yes	Yes	Yes	Yes
City x Language			Yes	Yes	Yes
Language x Time				Yes	Yes
City x Time					Yes
Adjusted R^2	0.284	0.284	0.284	0.287	0.288
Observations	2,238,606	2,238,606	2,238,606	2,238,606	2,238,606

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

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

Trends in programming languages and technologies or productivity shocks for certain programming languages and technologies captured by programming language $\times$ time fixed effects, decreases the coefficient for log size from 0.2524 (0.1429) to 0.1887 (0.0766) in column four of Table 1. The coefficients in columns three and four are statistically significant at the ten and one percent level, respectively. The decrease in the elasticity after adding controls for time trends in programming languages hints at larger clusters experiencing more positive productivity shocks.

When including all controls, i.e. also the dummies for city $\times$ time, the estimated elasticity is 0.2402 (0.1134) 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 2.4 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. A user's number of commits in technology one would increase by 3.87 percent, if she moves from the median cluster in technology four, Fresno-Madera, to the largest cluster, San Jose-San Francisco-Oakland. 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.

Though, the results have to be taken with caution. We only observe commits to public projects. If a user's cluster increases, she might move to committing more to private projects. Therefore, the results may provide a lower bound of the elasticity. [39]

5.2 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 with commits to projects with at least 100 star and only consider those commits. 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.[40] Table 2 shows the results of a regression of log commit on log size with the restricted sample.

The coefficient in the first column is insignificant with 0.2101 (0.2233). In the next columns it becomes significant at the ten percent level, varying from 0.4700 (0.2815) to 0.5968 (0.3300). In the final column, conditional on all fixed effects, the elasticity between commits and cluster size is significant at the ten percent level with 0.5968 (0.3298).

The positive elasticity between cluster size and number of commits of 0.5968 (0.3298) implies that a user commits 5.97 percent more in a technology to projects with at least 100 stars if her cluster in that technology increases by ten percent. This suggests a positive impact of cluster size on the quality of commits.

5.3 Heterogeneity in Elasticity

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. Contrary, 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

[39]Using the absolute cluster size instead of cluster density, results in exactly the same elasticity of 0.2402 (0.1134) with all covariates included. Additionally, we test if the results are stable for excluding large commits and large projects, i.e. commits bigger than 100 and projects with more than 40 users committing to. The elasticity slightly increases to 0.2490 (0.1103) and remains significant at the five percent level. See Table 26 and 27 in the Appendix for the regression results, respectively.

[40]These projects represent the upper 6% in the distribution of stars per project.

Table 2: Baseline Estimates - Excluding Projects with less than 100 Project Stars.

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.2101	0.4700*	0.4710*	0.3823*	0.5968*
	(0.2233)	(0.2815)	(0.2832)	(0.2251)	(0.3298)
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
City x Technology		Yes	Yes	Yes	Yes
City x Language			Yes	Yes	Yes
Language x Time				Yes	Yes
City x Time					Yes
Adjusted R^2	0.419	0.422	0.422	0.424	0.422
Observations	73,926	73,926	73,926	73,926	73,926

*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 at least 100 stars.

a smaller cluster.[41] 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 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 0.22362 (0.1122) in the smallest clusters, conditional on all covariates.[42] The estimates for different size quartiles range from 0.2362 (smallest size quartile; 0.1122) to 0.2146 (largest quartile; 0.1224), hence the variation in elasticities across quartiles is rather small.

Table 3: Heterogeneity in Elasticity by Cluster Size

	Log(Commit)
	(1)
First Quartile (Smallest)	0.2362**
	(0.1122)
Second Quartile	0.2318**
	(0.1132)
Third Quartile	0.2350**
	(0.1154)
Fourth Quartile (Largest)	0.2146*
	(0.1224)
Adjusted R^2	0.288
Observations	2,238,605
Wald (joint nullity), p-value	0.281

*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.

They suggest a mildly linear elasticity between cluster size and productivity, as the estimates decrease with size quartile. A Wald test for testing that all coefficients are zero in the model with all controls included, cannot be rejected with a p-value of 0.281. 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 does not find a heterogeneity in elasticity by cluster size.

Project Productivity Another cause of heterogeneity in elasticity could be project productivity. More productive projects, i.e., projects receiving a higher number of commits, might benefit

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

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

more from productivity spillovers. Findings suggest that in the case of firms, more productive firms gain the most from agglomeration effects (Combes et al., 2012).

Table 4: Heterogeneity in Elasticity by Project Productivity

	Log(Commit)
	(1)
First Quartile (Least Productive)	0.2320**
	(0.1105)
Second Quartile	0.2366**
	(0.1115)
Third Quartile	0.2341**
	(0.1122)
Fourth Quartile (Most Productive)	0.2402**
	(0.1150)
Adjusted R^2	0.288
Observations	2,238,606
Wald (joint nullity), p-value	0.293

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

Notes: Standard errors are clustered by city x technology. Project productivity is measured by the total number of commits a project received. The most productive projects are projects that are in the fourth quartile of the distribution of total commits per project. 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.

For GitHub projects, the same might be the case. Projects receiving a large number of commits likely have a higher quality demand on their received commits. They profit more if contributors acquire more knowledge from other users. In that case, we would expect an increase in the elasticity with respect to project productivity, measured by the project's total number of commits received.

Table 4 presents the estimates of a regression of log commit on log size, letting the slope of log size vary by project productivity quartile. All controls from the baseline estimation are included. The estimates for the lower three quartiles are smaller than the baseline estimate of 0.2402 (0.1134), with all covariates included. They are between 0.2320 (least productive; 0.1105), 0.2366 (second quartile; 0.1115) and 0.2341 (third quartile; 0.1122) and all significant at the five percent level. The elasticity for the most productive projects is also significant at the five percent level and slightly larger than the baseline estimate with 0.2404 (0.1150). A Wald test for testing that all coefficients are zero cannot be rejected with a p-value of 0.293.

This shows, that the elasticity between cluster size and number of commits increases with project productivity. The projects in the fourth quartile might be work projects and therefore experience a larger increase in commits with an increase in cluster size. However, based on the Wald test, we cannot reject the null hypothesis of no heterogeneity in cluster size with project productivity.

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 larger productivity increase with an increase in cluster size.

Table 5: Heterogeneity in Elasticity by Project Age

	Log(Commit)
	(1)
First Quartile (Youngest)	0.2241**
	(0.1107)
Second Quartile	0.2307**
	(0.1138)
Third Quartile	0.2392**
	(0.1153)
Fourth Quartile (Oldest)	0.2557**
	(0.1149)
Adjusted R^2	0.288
Observations	2,238,606
Wald (joint nullity), p-value	0.040

*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 × programming language, programming language × time, city × technology, technology, city × 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 5 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 0.2241 (youngest quartile; 0.1107) to 0.2557 (oldest quartile; 0.1149), 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 0.04. Thus, the elasticity between cluster size and productivity varies with project age. Especially for the oldest projects, the elasticity is the largest, and potentially knowledge spillovers matter the most.

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.[43]

Business Projects A last characteristics of projects which may lead to variation in cluster size is, if it is a leisure or work project. The latter might be more established, as it is for, or by, a firm and thus, requires contributors to be more knowledgeable about the project. 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 thus the project) benefits more from a larger cluster size as the users can, by exchanging knowledge, implement the acquired knowledge in their leisure project.

Table 6: Heterogeneity in Elasticity by Share of Commits made during Business Hours

	Log(Commit)
	(1)
First Quartile (Leisure)	0.2429**
	(0.1121)
Second Quartile	0.2457**
	(0.1143)
Third Quartile	0.2431**
	(0.1134)
Fourth Quartile (Business)	0.2101*
	(0.1135)
Adjusted R^2	0.288
Observations	2,238,606
Wald (joint nullity), p-value	0.020

*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.

[43]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.

In Table 6 we let the elasticity between cluster size and productivity vary by the share of commits 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 are the projects that are in the fourth quartile of share of business commits per project.

The elasticity, conditional on all controls from the baseline model, is very similar for the first three quartiles with 0.2429 (first quartile; 0.1121), 0.2458 (second quartile; 0.1143) and 0.2431 (third quartile; 0.1134) 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 0.2101 (0.1134) 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 0.02. 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.

Alternative User Samples Lastly, we estimate the elasticity between cluster size and the number of commits for different user samples. In detail, we calculate a user's share of commits to 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, i.e., higher in the distribution of total commits per user, might benefit more from larger clusters. Table 7 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 0.5437 (0.2776) and significant at the ten percent level. The coefficient for the upper 50 percent is 0.2866 (0.1468) and also significant at the ten percent level. For the upper 75 percent of users, the elasticity is significant at the five percent level with 0.2484 (0.1192) and similar to the baseline estimate of 0.2402 (0.1134).

The largest elasticity for the upper 25 percent might match up with the results for heterogeneity in elasticity with respect to project productivity. Users in the highest quartile might use GitHub for work reasons, hence a change in cluster size might increase their commits the most. Users in the lower quartiles possibly use GitHub more for private projects, which explains the slightly smaller elasticity. This is only hypothetically, which cannot be tested by the data on hand and requires more knowledge on the users themselves and the projects they commit to.[44]

[44]With more data on users on hand, one could also investigate if gender plays a role on the effects of cluster size on productivity. Rosenthal and Strange (2012) find smaller agglomeration effects for women than for men. Similarly might be the case for female users. They possibly benefit less from larger clusters than male users.

Table 7: Alternative User Samples

	Log(Commit)			
	Upper 25%	Upper 50%	Upper 75%	All
	(1)	(2)	(3)	(4)
Log(Size)	0.5437*	0.2866*	0.2484**	0.2402**
	(0.2776)	(0.1468)	(0.1192)	(0.1134)
Adjusted R^2	0.382	0.321	0.294	0.288
Observations	766,968	1,589,540	2,115,306	2,238,606

*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.

5.4 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.

First, a large share of projects is very localized, with all users stemming from only one city.[45] Splitting the sample by projects, where all users stem from one city vs. more geographically distributed projects shows, the effect for local projects with 0.2234 (0.0836) is almost identical to our baseline elasticity and increases in significance to the one percent level. For more distributed projects, there is a slightly smaller, but still significant effect of 0.1926 (0.1144). It seems, however, that especially local projects drive our results as they make up a large share of observations and the effect increases in significance. This supports also the hypothesis of knowledge spillovers leading to an increase in activity. Especially, when users are physically close to each other, knowledge can flow more easily via face-to-face interactions (Atkin et al., 2022) and team communication is improved (Ayoubi et al., 2017).[46] This seems also to be the case for OSS contributors.

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 0.2336 (0.1171).[47] For larger projects, i.e. with more than 40 users, the effect almost triples to 0.7292 (0.3385) and is still significant at the five percent level. As Ayoubi

[45]In 75% of projects, all users stem from one city. See Table 23 for more details.

[46]Atkin et al. (2022) show with granular smartphone geolocation data for workers of firms in the Silicon Valley that face-to-face interactions measured by meetings has a positive effect on knowledge flows, i.e. patent citation. With our estimate mainly stemming from local projects, face-to-face interactions may also be one cause of the observed positive elasticity between productivity and cluster size.

[47]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.

Table 8: Local and More Distributed Projects

	Log(Commit)	
	Distributed	Local
	(1)	(2)
Log(Size)	0.1926*	0.2234***
	(0.1143)	(0.0836)
Adjusted R^2	0.364	0.279
Observations	778,668	1,459,938

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

Notes: Standard errors are clustered by city x technology. Every column presents a regression. Locality of a project is measured by the share of users stemming from different cities. A project with a share of 1 means, that all users committing to the project stem from one city. The sample is split by 1, i.e. projects with all users stemming from one city in comparison to projects with more geographically distributed users 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.

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.[48]

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 personnel 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.

Considering commits to work in contrast to leisure projects show, that the baseline estimates is mainly stemming from leisure projects. We consider a project as a work project if all commits received are during business hours.[49] Almost one quarter of projects in our sample have a share

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

[49]Business hours are Monday through Friday from 6am to 8pm, similarly implemented by McDermott and Hansen (2021).

Table 9: Small and Large Projects

	Log(Commit)	
	Small	Large
	(1)	(2)
Log(Size)	0.2336**	0.7292**
	(0.1171)	(0.3385)
Adjusted R^2	0.299	0.411
Observations	2,176,020	62,586

*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.

Table 10: Others and Own Projects

	Log(Commit)	
	Others	Own
	(1)	(2)
Log(Size)	0.2314*	0.0923
	(0.1343)	(0.1471)
Adjusted R^2	0.320	0.312
Observations	1,261,215	977,391

*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.

of business commits of one.[50]. However, leisure projects seem to be driving our main results more than business projects. A large share of OSS contributors report to not being paid for working on OSS projects, however contributions are done during work hours. Research suggests that this is to a large extent done without the knowledge of the employer (Lakhani and Wolf, 2003). The findings of Table 11 are in line with the results of the heterogeneity analysis with respect to business quartiles in Table 6. For projects with a smaller share of business commits the elasticity was larger than for work projects.

Table 11: Leisure and Business Projects

	Log(Commit)	
	Leisure	Business
	(1)	(2)
Log(Size)	0.2366**	0.2840
	(0.1102)	(0.4828)
Adjusted R^2	0.327	-0.389
Observations	1,752,610	485,996

*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 share of commits during business hours, i.e. Monday through Friday from 6am to 8pm was calculated. A project is identified as a business project with a share of 1, all commits are made during business hours. 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.

For some users, GitHub may be a way of labor market signalling. By contributing to projects on the platform they can show their skills and attract employers. This is likely done through leisure projects.

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.

5.5 Instrumental Variable Estimates - 2SLS

To address endogeneity concerns 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, biasing the estimated elasticity, should be removed. For the regression we focus on the most skilled users, i.e. users in the top quartile

[50]See Table 23 for a summary of the business share per projects in our regression sample.

in the follower per user distribution. First of all, these users are more active and observable in consecutive time intervals with commits to projects. Second, for them the gains in productivity may be more precisely measured. The number of followers per user can be seen as a measure of skills. The higher the number of followers the more skilled the user likely is (Lee et al., 2013).

For comparison, Table 12 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.

Table 12: First Differences Estimates - Full Sample

	Δ Log(Commit)		
	(1)	(2)	(3)
Δ Log(Size)	0.0138	0.0154	0.0154
	(0.0097)	(0.0181)	(0.0181)
Fixed-effects			
Language x Time	Yes	Yes	Yes
City x Time	Yes	Yes	Yes
Project		Yes	Yes
Language			Yes
Observations	290,363	290,363	290,363

*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).

For them, the estimated coefficients are positive, but statistically insignificant and small in magnitude. Hence, no clear relationship between a change in log cluster size and change in log commits can be estimated for the full sample with this approach.

In Table 13 the first differences model is estimated with a sample restricted to users in the top quartile of the follower per user distribution.

In this case, the estimate reflects the contemporaneous change in productivity with a change in cluster size. The coefficient of cluster size is smaller and positive with 0.0956 (0.0500) in column three, conditional on programming language $\times$ time, city $\times$ time, project and programming language fixed effects.[51] It is significant at the ten percent level. The smaller and positive estimate in comparison to the baseline estimate with 0.2402 (0.1134) is partly due to its representation of the contemporaneous effect of cluster size on productivity, excluding effects of knowledge spillovers

[51]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 and city $\times$ programming language might be too conservative.

Table 13: First Differences Estimates - IV Sample

	Δ Log(Commit)		
	(1)	(2)	(3)
Δ Log(Size)	0.0341**	0.0956*	0.0956*
	(0.0159)	(0.0500)	(0.0500)
Fixed-effects			
Language x Time	Yes	Yes	Yes
City x Time	Yes	Yes	Yes
Project		Yes	Yes
Language			Yes
Observations	68,694	68,694	68,694

*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 and users in the upper fourth quartile of the follower per user distribution. 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).

that increase productivity with a delay.[52] 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 0.89009 (0.43531) and significant at the five percent level as presented in Table 14, column three. The first stage in column three is significant at the one percent level with an estimate of 0.0002 (0.00001) conditional on all covariates.[53] Changes in cluster size elsewhere are a good predictor of local cluster size, however, size-wise relatively small. The instrument is very strong with an F-test of 73.81. A Wu-Hausmann test gives a p-value of 0.12, i.e., the absence of endogeneity in the instrumented variable, the change in cluster size, can be rejected.

[52]The estimate does not seem to be driven by the now smaller sample. Estimating the baseline model with the first differences sample results in a negative and insignificant elasticity of -0.2003 (0.4737). Results are shown in Table 28 in the Appendix.

[53]In a reduced form estimate, i.e., using the changes in other projects elsewhere to predict the changes in commits, it results in a significant estimate of 0.0002 (0.00001) presented in Table 29 in the Appendix.

Table 14: 2SLS Estimates

	Δ Log(Commit) (1)	Δ Log(Commit) (2)	Δ Log(Commit) (3)
First Stage	0.00002***	0.00002***	0.00002***
	(0.00001)	(0.00001)	(0.00001)
Δ Log(Size)	0.85540**	0.89009**	0.89009**
	(0.38402)	(0.43522)	(0.43531)
Fixed-effects			
Language x Time	Yes	Yes	Yes
City x Time	Yes	Yes	Yes
Project		Yes	Yes
Language			Yes
Observations	68,694	68,694	68,694
F-test (1st stage)	62.86	73.84	73.81
Wu-Hausman, p-value	0.07	0.12	0.12

*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 and users in the upper fourth quartile of the follower per user distribution. 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).

5.6 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 15 presents the regression results for different subsamples with all covariates from the baseline model included.

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 insignificant in the first four columns and decreases from 0.1511 (0.0976)

Table 15: Different Lengths of Observation Period

	Log(Commit)				
	1	2	3	4	5
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1511	0.1507	0.1411	0.1455	0.1712*
	(0.0976)	(0.0972)	(0.0936)	(0.0925)	(0.0907)
Adjusted R^2	0.178	0.184	0.226	0.245	0.255
Observations	5,677,085	5,640,879	4,761,734	4,180,116	3,655,988

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

Table 16: Different Lengths of Observation Period - Continued

	Log(Commit)				
	6	7	8	9	10
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.1842**	0.1735*	0.1870*	0.2080**	0.2402**
	(0.0921)	(0.0938)	(0.0966)	(0.1011)	(0.1135)
Adjusted R^2	0.262	0.267	0.273	0.278	0.288
Observations	3,355,702	3,119,287	2,873,106	2,656,882	2,238,606

*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.

to 0.1455 (0.0925). This indicates that the sample of users with non-zero commits between at least one to at least four time intervals contains users for which no significant relationship between cluster size and productivity can be found. One reason might be, due to their lower activity on GitHub, productivity changes cannot be observed, or at least not in a statistical sense.

The coefficients increase in size in the following columns five to ten, as well as becoming significant at the ten percent level (column five, seven and eight) to five percent level (columns six, nine and ten). The elasticity in column ten, which presents the baseline estimate, is the largest in size with 0.2402 (0.1135). Hence, the assumption of 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 the bias towards zero should be the least pronounced. 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. By definition, clusters calculated only with active users is smaller, as it contains only users that commit.

Table 17 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 0.0935 (0.0802) but insignificant. After adding further controls for city $\times$ programming language, city $\times$ technology, programming language $\times$ time, and city $\times$ time, the estimate increases to 0.1653 (column six; 0.0638) and is significant at the one 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 1.65 percent. This implies a significant raise in the number of commits as a result of more local active users. It seems that other local active users positively influence a user's activity.

In comparison to the baseline estimate with 0.2402 (0.1134), the elasticity becomes smaller

Table 17: Active Users

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Active Users)	0.0935	0.1464*	0.1448*	0.1087**	0.1653***
	(0.0802)	(0.0754)	(0.0763)	(0.0488)	(0.0638)
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
City x Technology		Yes	Yes	Yes	Yes
City x Language			Yes	Yes	Yes
Language x Time				Yes	Yes
City x Time					Yes
Adjusted R^2	0.284	0.284	0.284	0.287	0.288
Observations	2,238,104	2,238,104	2,238,104	2,238,104	2,238,104

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

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

when restricting cluster calculation to only active users but increases in significance. The smaller elasticity may be due to underestimating cluster size as the inactive users are not considered. Though, again, it still could be that users are active by committing to private projects which we are not able to observe.

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

6 Conclusion

We find a significant elasticity of 0.2402 (0.1134) between productivity and cluster size for GitHub users conditional on several controls. Regarding the quality of commits, estimated by the number of commits to projects with at least 100 star it is positive with 0.1245 (0.0642). 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 low-skilled users, i.e. users with a small number of followers, benefit the most from changes in cluster size.

Selection concerns seem to play a minor role as our dynamic response and event study analysis suggests. Future values of cluster size are only up to one period ahead correlated with current productivity. Larger clusters seem not to systematically attract users with increasing productivity.

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.

Nevertheless, the results suggest that productivity spillover effects and cluster size play a role in open source software development as well. They are an important input source for firms and, thus, fostering knowledge creation would further increase the benefits from integrating open source software. Especially in times of increasing working from home, cities and urban density keep playing an important role in the diffusion of knowledge.

References

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

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

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

Audretsch, D. and Feldmann, M. (1996). R&D spillovers and the geography of innovation and production. *American Economic Review*, 86:630–640.

Autor, D. H., Levy, F., and Murnane, R. J. (2003). The skill content of recent technological change: An empirical exploration. *The Quarterly journal of economics*, 118(4):1279–1333.

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

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

Bacolod, M., Blum, B. S., and Strange, W. C. (2009). Skills in the city. *Journal of Urban Economics*, 65(2):136–153.

Baum-Snow, N., Gendron-Carrier, N., and Pavan, R. (2020). Local productivity spillovers.

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

Belenzon, S. and Schankerman, M. (2008). Motivation and sorting in open source software innovation. *CEPR Discussion Papers*, 7012.

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

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

Borjas, G. J. and Doran, K. B. (2015). Which peers matter? The relative impacts of collaborators, colleagues, and competitors. *Review of economics and statistics*, 97(5):1104–1117.

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

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

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

Casalnuovo, C., Vasilescu, B., Devanbu, P., and Filkov, V. (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*, pages 817–828.

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

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

Cohen, W., Nelson, R., and Walsh, J. (2000). Protecting their intellectual assets: appopriability conditions and why u.s. manufcaturing firms patent (or not). *National Bureau of Economic Research Working Paper*, (w7552).

Combes, P.-P., Duranton, G., Gobillon, L., Puga, D., and Roux, S. (2012). The productivity advantages of large cities: Distinguishing agglomeration from firm selection. *Econometrica*, 80(6):2543–2594.

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

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

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

Demsas, J. (2021). Remote work is overrated. America's supercities are coming back. *Vox*.

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

Emanuel, N. and Harrington, E. (2020). "working" remotely?

Fackler, T. and Laurentsyeva, N. (2020). Gravity in online collaborations: Evidence from GitHub. *CESifo Forum*, 21(3).

Farhauer, O. and Kröll, A. (2009). Die shift-share-analyse als instrument der regional-und clusterforschung. Technical report, Passauer Diskussionspapiere-Volkswirtschaftliche Reihe.

Foster, N. and Stehrer, R. (2009). Sectoral productivity, density and agglomeration in the wider europe. *Spatial Economic Analysis*, 4(4):427–446.

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, pages 233–236, Piscataway, NJ, USA. IEEE Press.

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

Herbst, D. and Mas, A. (2015). Peer effects on worker output in the laboratory generalize to the field. *Science*, 350(6260):545–549.

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

Hindle, A., German, D. M., and Holt, R. (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*, pages 99–108.

Jaffe, A. B., Trajtenberg, M., and Henderson, R. (1993). Geographic localization of knowledge spillovers as evidenced by patent citations. *The Quarterly Journal of Economics*, 108.

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

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

Lakhani, K. R. and Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *Open Source Software Projects (September 2003)*.

Laurentsyeva, N. (2019). From friends to foes: National identity and collaboration in diverse teams. *CRC TRR 190 Discussion Paper*, 226.

Lee, M. J., Ferwerda, B., Choi, J., Hahn, J., Moon, J. Y., and Kim, J. (2013). Github developers use rockstars to overcome overflow of news. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 133–138.

Lima, A., Rossi, L., and Musolesi, M. (2014). Coding together at scale: GitHub as a collaborative social network. *arXiv preprint arXiv:1407.2535*.

Manso, G. (2011). Motivating innovation. *The Journal of Finance*, 66(5):1823–1860.

Marshall, A. (1890). Principles of economics macmillan. *London (8th ed. Published in 1920)*.

McDermott, G. R. and Hansen, B. (2021). Labor reallocation and remote work during covid-19: Real-time evidence from github.

McDonald, N. and Goggins, S. (2013). Performance and participation in open source software on GitHub. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, pages 139–144. CHI EA '13, Paris, France.

Melo, M. T., Nickel, S., and Saldanha-Da-Gama, F. (2009). Facility location and supply chain management–a review. *European journal of operational research*, 196(2):401–412.

Microsoft (2021). Microsoft to acquire github for $7.5 billion. `https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/`. Accessed: 2021-06-01.

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.

Pischke, S. (2007). Lecture notes on measurement error.

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

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 Strange, W. C. (2012). Female entrepreneurship, agglomeration, and a new spatial mismatch. *Review of Economics and Statistics*, 94(3):764–788.

Rosenthal, S. S. and Strange, W. C. (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.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Influence of social and technical factors for evaluating contribution in GitHub. In *Proceedings of the 36th international conference on Software engineering*, pages 356–366.

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., and Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816.

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

Wright, N., Nagle, F., and Greenstein, S. M. (2020). Open source software and global entrepreneurship. *Harvard Business School Technology & Operations Mgmt. Unit Working Paper*, (20-139):20–139.

Wuchty, S., Jones, B. F., and Uzzi, B. (2007). The increasing dominance of teams in production of knowledge. *Science*, 316(5827):1036–1039.

A Appendix

A.1 Tables

Table 18: Summary Statistics Commits by Programming Languages - Full Data

Language	Min.	Median	Mean	Max.	Projects	N	Commits	Share
C	1	8	89.16	77,072	174,590	58,993	5,259,919	4.1%
C#	1	10	70.76	18,463	161,317	49,141	3,477,377	2.71%
C++	1	10	94.67	56,647	212,458	72,488	6,862,092	5.35%
CSS	1	18	71.53	225,948	882,948	242,998	17,382,342	13.55%
Go	1	10	109.85	26,611	151,212	37,262	4,093,291	3.19%
Java	1	11	78.00	221,308	426,890	118,240	9,223,283	7.19%
JavaScript	1	25	122.04	364,996	1,672,610	244,945	29,893,000	23.3%
Jupyter Notebook	1	10	57.83	25,569	154,538	45,260	2,617,263	2.04%
Objective-C	1	8	49.96	9,075	77,691	24,744	1,236,288	0.96%
PHP	1	10	92.18	218,260	210,940	56,839	5,239,254	4.08%
Python	1	14	95.12	94,083	699,597	176,131	16,753,689	13.06%
R	1	9	72.85	73,039	72,453	21,232	1,546,843	1.21%
Ruby	1	18	163.76	73,674	834,972	89,742	14,696,271	11.45%
Rust	1	11	104.94	67,676	44,368	11,521	1,209,069	0.94%
Shell	1	9	63.42	281,997	184,134	73,468	4,659,638	3.63%
Swift	1	9	49.49	30,077	86,336	23,894	1,182,525	0.92%
TypeScript	1	8	59.46	20,600	141,772	50,025	2,974,364	2.32%

Table 24: Largest Clusters for Technologies in 202103 Snapshot

	Size
1	
San Jose-San Francisco-Oakland, CA	0.12554
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.09891
Seattle-Tacoma-Olympia, WA	0.06455
Los Angeles-Long Beach-Riverside, CA	0.05337
Toronto	0.04500
Boston-Worcester-Manchester, MA-NH	0.03885
Chicago-Naperville-Michigan City, IL-IN-WI	0.03311
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03121
Austin-Round Rock, TX	0.02827
Denver-Aurora-Boulder, CO	0.02790
2	

San Jose-San Francisco-Oakland, CA	0.15369
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.09970
Seattle-Tacoma-Olympia, WA	0.06551
Boston-Worcester-Manchester, MA-NH	0.05008
Los Angeles-Long Beach-Riverside, CA	0.04798
Toronto	0.03935
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03576
Chicago-Naperville-Michigan City, IL-IN-WI	0.03273
Denver-Aurora-Boulder, CO	0.02775
Austin-Round Rock, TX	0.02720
3	
San Jose-San Francisco-Oakland, CA	0.15546
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.12714
Seattle-Tacoma-Olympia, WA	0.05215
Los Angeles-Long Beach-Riverside, CA	0.04429
Chicago-Naperville-Michigan City, IL-IN-WI	0.04422
Boston-Worcester-Manchester, MA-NH	0.04128
Denver-Aurora-Boulder, CO	0.04110
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.03371
Toronto	0.03338
Portland-Vancouver-Beaverton, OR-WA	0.02721
4	
San Jose-San Francisco-Oakland, CA	0.16147
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.09112
Seattle-Tacoma-Olympia, WA	0.06347
Los Angeles-Long Beach-Riverside, CA	0.05121
Toronto	0.04519
Boston-Worcester-Manchester, MA-NH	0.04058
Chicago-Naperville-Michigan City, IL-IN-WI	0.03382
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.02891
Austin-Round Rock, TX	0.02534
Atlanta-Sandy Springs-Gainesville, GA-AL	0.02354
5	
San Jose-San Francisco-Oakland, CA	0.16001
New York-Newark-Bridgeport, NY-NJ-CT-PA	0.07661

Seattle-Tacoma-Olympia, WA	0.07520
Los Angeles-Long Beach-Riverside, CA	0.05180
Boston-Worcester-Manchester, MA-NH	0.04500
Toronto	0.03707
Denver-Aurora-Boulder, CO	0.02692
Austin-Round Rock, TX	0.02662
Chicago-Naperville-Michigan City, IL-IN-WI	0.02653
Washington-Baltimore-Northern Virginia, DC-MD-VA-WV	0.02653

Table 19: Summary Statistics Commits by Programming Languages - Regression Data

Language	Min.	Median	Mean	Max.	Projects	N	Commits	Share
C	1	19	340.67	60,823	64,278	7,510	2,558,443	6.41%
C#	1	20	322.03	18,116	30,697	3,473	1,118,404	2.8%
C++	1	23	364.61	56,647	57,220	7,557	2,755,341	6.9%
CSS	1	79	276.28	225,948	155,878	15,840	4,376,271	10.97%
Go	1	20	290.65	24,648	54,765	5,561	1,616,284	4.05%
Java	1	25	392.21	221,308	87,435	8,392	3,291,425	8.25%
JavaScript	1	128	519.07	172,663	323,606	15,571	8,082,383	20.25%
Jupyter Notebook	1	17	116.88	9,212	12,299	2,723	318,267	0.8%
Objective-C	1	10	118.05	9,075	17,207	3,176	374,918	0.94%
PHP	1	23	357.26	218,260	56,859	6,065	2,166,770	5.43%
Python	1	64	471.43	41,771	167,390	12,884	6,073,841	15.22%
R	1	27	415.98	73,039	17,409	1,551	645,181	1.62%
Ruby	1	36	354.62	48,734	127,808	9,412	3,337,658	8.36%
Rust	1	19	218.47	41,287	16,602	2,326	508,171	1.27%
Shell	1	26	150.57	23,715	56,690	10,943	1,647,734	4.13%
Swift	1	15	154.21	30,077	11,639	1,857	286,362	0.72%
TypeScript	1	15	142.87	20,600	26,873	5,224	746,332	1.87%

Table 20: Summary Statistics Commits by Technologies - Full Data

Technology	Min.	Median	Mean	Max.	Projects	N	Commits	Share
1	1	42	175.80	365,159	3,069,587	335,409	58,966,337	45.96%
2	1	19	128.03	284,363	1,261,934	231,750	29,670,724	23.12%
3	1	18	163.76	73,674	834,972	89,742	14,696,271	11.45%
4	1	13	82.49	221,308	590,917	141,136	11,642,096	9.07%
5	1	12	123.28	171,799	431,416	108,137	13,331,080	10.39%

Table 21: Summary Statistics Commits by Technologies - Regression Data

Technology	Min.	Median	Mean	Max.	Projects	N	Commits	Share
1	1	337	937.85	235,565	593,913	17,583	16,490,160	41.32%
2	1	143	651.90	73,530	308,553	15,802	10,301,307	25.82%
3	1	36	354.62	48,734	127,808	9,412	3,337,658	8.36%
4	1	32	398.38	221,308	116,281	9,922	3,952,705	9.91%
5	1	40	549.60	60,884	138,100	10,593	5,821,955	14.59%

Table 22: Summary Statistics of Commits, Projects and Users - Full Data

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Length User Observed	1	5	9	7.46	10	10
Commits per User	1	25	78	317.08	236	388,287
Commit per Project per Snapshot	1	1	3	14.20	9	364,392
Stars per Project	0	0	0	29.30	0	259,118
Stars per Project - Star $>$ 0 and non-forked Projects	1	1	4	171.18	24	259,118
Forks per Project	0	0	0	3.56	0	145,997
Forks per Project - Forks $>$ 0 and non-forked Projects	1	1	2	28.80	7	145,997
Programming Language per City	3	16	18	16.45	18	18
Programming Language per City per Snapshot	1	8	13	12.02	16	18
Technology per City	1	5	5	4.87	5	5
Technology per City per Snapshot	1	4	5	4.27	5	5
Programming Language per User	1	3	4	4.25	5	18
Programming Language per User per Snapshot	1	1	2	2.12	3	17
Technology per User	1	1	2	2.24	3	5
Technology per User per Snapshot	1	1	1	1.68	2	5
Own Project	0	0	1	0.73	1	1
Business Share	0	0	1	0.59	1	1
Weekend Share	0	0	0	0.21	0	1
Out of Hour Share	0	0	0	0.32	1	1
Local Share	0	1	1	0.97	1	1
Users per Project	1	1	1	1.24	1	324,321
Project Age (in Years)	0	2	4	3.97	6	13

Table 23: Summary Statistics of Commits, Projects and Users - Regression Data

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Length User Observed	10	10	10	10.00	10	10
Commits per User	25	536	1,089	2,171.40	2,336	235,640
Commit per Project per Snapshot	1	1	3	17.83	10	169,209
Stars per Project	0	0	0	88.01	2	259,118
Stars per Project - Star > 0 and non-forked Projects	1	2	9	295.29	58	259,118
Forks per Project	0	0	0	11.22	0	145,997
Forks per Project - Forks > 0 and non-forked Projects	1	1	3	53.33	14	145,997
Programming Language per City	1	11	16	13.28	17	17
Programming Language per City per Snapshot	1	5	10	9.46	14	17
Technology per City	1	5	5	4.55	5	5
Technology per City per Snapshot	1	3	4	3.62	5	5
Programming Language per User	1	5	6	6.53	8	17
Programming Language per User per Snapshot	1	2	3	3.25	4	16
Technology per User	1	3	4	3.45	4	5
Technology per User per Snapshot	1	1	2	2.26	3	5
Own Project	0	0	1	0.50	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	1.66	1	2,145
Project Age (in Years)	0	3	5	5.00	7	13

Table 25: Summary Statistics - Clusters 202103 Snapshot

Technology	10th Perc.	Median	90th Perc.	Max.
1	0.00003	0.00049	0.00894	0.12554
2	0.00001	0.00042	0.00866	0.15369
3	0.00004	0.00057	0.01032	0.15546
4	0.00002	0.00046	0.01104	0.16147
5	0.00002	0.00056	0.01049	0.16001

Table 26: Absolute Cluster Size

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Absolute Cluster Size)	-0.2491***	-0.2920***	-0.2924***	0.1887**	0.2402**
	(0.0659)	(0.0760)	(0.0763)	(0.0766)	(0.1134)
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
City x Technology		Yes	Yes	Yes	Yes
City x Language			Yes	Yes	Yes
Language x Time				Yes	Yes
City x Time					Yes
Adjusted R^2	0.284	0.285	0.285	0.287	0.288
Observations	2,238,606	2,238,606	2,238,606	2,238,606	2,238,606

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

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

Table 27: Baseline Estimates - Excluding Projects with large Commits and large Projects

	Log(Commit)				
	(1)	(2)	(3)	(4)	(5)
Log(Size)	0.0932	0.2228	0.2213	0.1533*	0.2490**
	(0.1089)	(0.1485)	(0.1497)	(0.0850)	(0.1103)
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
City x Technology		Yes	Yes	Yes	Yes
City x Language			Yes	Yes	Yes
Language x Time				Yes	Yes
City x Time					Yes
Adjusted R^2	0.253	0.254	0.254	0.257	0.258
Observations	2,113,098	2,113,098	2,113,098	2,113,098	2,113,098

*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 28: Baseline Estimates - IV Sample

	Log(Commit)
	(1)
Log(Size)	-0.2003
	(0.4737)
Fixed-effects	
City	Yes
Time	Yes
Language	Yes
Technology	Yes
Project	Yes
User	Yes
City x Technology	Yes
City x Language	Yes
Language x Time	Yes
City x Time	Yes
Adjusted R^2	0.501
Observations	68,694

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

Notes: Standard errors are clustered by city x technology. Every column presents a regression. The sample consists of commits to projects, that receive commits in two consecutive time intervals and users in the upper fourth quartile of the follower per user distribution.

Table 29: 2SLS Estimates - Reduced Form

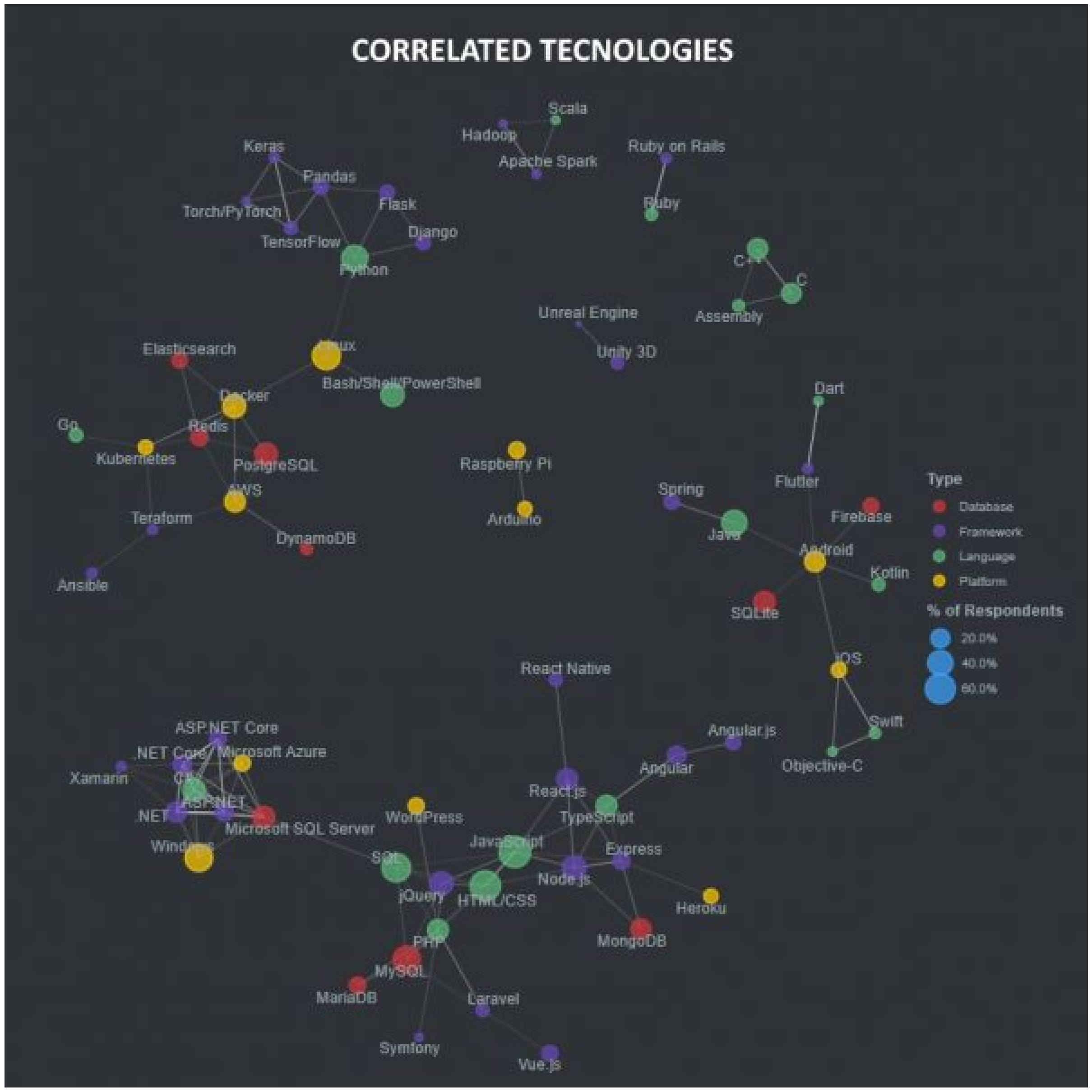
	Δ Log(Commit)		
	(1)	(2)	(3)
First Stage	0.00001***	0.00002*	0.00002*
	(0.00001)	(0.00001)	(0.00001)
Fixed-effects			
Language x Time	Yes	Yes	Yes
City x Time	Yes	Yes	Yes
Project		Yes	Yes
Language			Yes
Adjusted R^2	0.115	-0.005	-0.006
Observations	68,694	68,694	68,694

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

Notes: Standard errors are clustered by city x technology. Every column presents a regression. The sample consists of commits to projects, that receive commits in two consecutive time intervals and users in the upper fourth quartile of the follower per user distribution. 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).

A.2 Figures

Figure 7: Correlated Technologies by Stack Overflow Developer Survey 2020



Notes: Accessed via https://insights.stackoverflow.com/survey/2020#correlated-technologies.