

# When dispersed teams are more successful: Theory and evidence from software\*

Gábor Békés<sup>†</sup>, Julian Hinz<sup>‡</sup>, Miklós Koren<sup>§</sup> and Aaron Lohmann<sup>¶</sup>

13 September, 2024. Draft, v.0.4.1

## Abstract

Who works with whom when collaboration is not constrained by the need to be physically co-located? How does the success of projects depend on the geography of partners? We build a model of global team formation and collaboration. In the model, people have heterogeneous and partially observable skills, collaboration is costly across locations, and success depends on the best idea from a team. The model yields five testable implications: (I) collaboration is more likely among nearby persons, (II) there is selection, only highly skilled distant people form teams, (III) geographically diverse teams are more successful, (IV) for more complex projects, the diversity-success elasticity is higher, and (V) there is a non-linear relationship between team size and project quality. We use a very granular dataset of open-source software development to test these predictions with success linked to downstream usage of projects. The advantage of open-source software — next to its near total transparency of the production process — is the radical lack of physical needs that allows us to focus on interpersonal aspects only. Results shall generalize to other dispersed knowledge base activities.

**Keywords:** Open-source Software, collaboration, frictions, team formation, diversity

**JEL Classification:** F10, F14, L86, R12, O33

---

\*We are grateful for comments and suggestions to Gianmarco Ottaviano, Johannes Boehm and seminar participants at CEU, Bocconi, Torino, USI Lugano, Kiel, Bayreuth, Linz and EEA ETSG, FIW meetings. This work was funded by the European Union under the Horizon Europe grant 101061123. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

<sup>†</sup>Central European University, HUN-REN KRTK and CEPR.

<sup>‡</sup>Bielefeld University and Kiel Institute for the World Economy.

<sup>§</sup>Central European University, HUN-REN KRTK, CEPR and Cesifo.

<sup>¶</sup>Bielefeld University and Kiel Institute for the World Economy.

# 1 Introduction

The production of services, software development, and other knowledge-intensive activities are potentially subject to only minor geographic constraints due to modern communication technologies. Still, teams tend to be closely co-located simply by virtue of belonging to the same firm. What happens if that boundary also falls? How does the quality of projects interact with the geographic composition of the teams which produce them. We study this question by building a model of global team formation and testing its implications on a granular dataset of open source software developers from GitHub, the largest online collaboration tool for developing software. Our findings have implications for sourcing decisions within the services sector and contemporary work arrangements, such as freelance engagements. One of our main findings is that geographically dispersed teams are typically associated with higher quality products. This effect, we argue mostly driven by selection, is more important than the traditionally discussed effects of agglomeration economics. We see this finding as another piece of evidence of the importance of the extensive margin which is highlighted and emphasised by contributions in the international trade literature like Allen (2014), Chaney (2014) or more recently Eaton et al. (2022).

Our first contribution is to provide a model which evolves around heterogeneous developers with partially observable skills, costly collaboration across locations and a production function which depends on the best idea of team members. Project ideas have a general attractiveness which incentivise developers to contribute. Credit sharing between developers is however not equal mirroring first author bias in academia. The model yields five testable implications: (I) collaboration is more likely among nearby persons, (II) There is selection, only highly skilled distant people form teams, (III) geographically diverse teams are more successful, (IV) for more complex projects, diversity-success elasticity is higher, and (V) there is a non-linear relationship between team size and project quality.

Our second contribution lies in studying novel data and providing descriptive evidence. The empirical context to quantify the model predictions comes from the Open Source Software (OSS) industry. With software being everywhere in our digital world, open source software is almost always coming with it. Estimates of the importance of OSS are generally high – the Open Source Monitor 2023 found that over three quarters of German firms use OSS and over a half actively contribute. Recently, Hoffmann et al. (2024) estimated the demand value of OSS at \$8.8 trillion globally. The magnitudes of these estimates underscore the key role OSS plays and motivates the focus of this paper on this side of the software industry. With that being said, our data is vast, global in scale and granular. The main part of it comes GitHub.com which is the platform that has proven dominant for sharing and developing open source software and more particularly we rely on a data project GHtorrent (see Gousios and Spinellis (2012)). As of 2024 there are over 100 million users registered who work on more than 150 million software projects (see

GitHub innovation). With the size of the platform, GitHub also started to become a social network in which developers share information about themselves which we exploit to place developers in cities. The best way to understand our data is as a mix of project level development data paired with information on users. To be more precise, the raw data contains information on 72 million projects, coming from 18 million user IDs in the time period of 2012 till June of 2019. Furthermore, we can map 1,362,106 users into cities across the world. Even though this is a relatively small share of the total data, we show that this can be interpreted as representative by showing that our aggregated shares match overall aggregates of the industry. Given the rather unique data opportunities of these datasets also for future research, we discuss it in detail including descriptives in the data section.

A first inspection of developers reveals that most come from the Northern hemisphere with 21% located from the United States. Additionally, software development is strongly clustered with the most represented cities being San Francisco, London and New York and Beijing. However, we also observe strong activity in India and Brazil. Traditionally knowledge intensive activity in such countries is not represented quite as intensely due to data being tied to patenting offices. OSS can potentially be a pathway to also study knowledge intensive sectors in more countries and not be constrained by patenting offices.

In terms of collaboration, we observe that about 27% projects are developed collaboratively with at least two developers. After that there is a steep power law like decline. As an additional data source, we use Libraries.io. Libraries.io is a data effort which aims to collect linkages between open source software projects. It is common to rely on other software projects through so called dependencies. Essentially, this is an input statistics or can be interpreted as a usage statistics. Downstream libraries will be our preferred measure of success of open source projects. We will however also consider an alternative measure which comes directly from our Github data "Stars". On Github developers can indicate that they "like" a project by giving a star.

Our third contribution lies in quantifying the empirical predictions made earlier. Equipped with the data, we test the empirical predictions of our model. First, we study how distance effects collaboration with a standard gravity approach. To do so, we aggregate developer links – working on the same project – at a city to city level. Employing a PPML regression, we find sizable estimates for distance elasticities. The coefficient on same language also enters the regression positively as expected. Further, we zoom in deeper into this relationship and show that distance elasticities follow a non-linear shape with collaboration being highly concentrated in same city projects.

For the remaining regressions we build our dataset in a pseudodynamic version which contains three periods in time for each project. A before, current and after period. The before period is included to capture the prior experience of developers before starting the

project. The current period is included to capture the amount of work effort exerted in the period of production and finally the after period captures success outcomes.

Projects have static values like the programming language and the quarter in which they were initiated. The location of developers will also be treated as a static variable which does not change. By comparing developers who are collaborating with those who never enter collaboration, we show that those who collaborate have on average higher skill levels by exploring different measures. As our main analysis, we regress the discussed success variables on different distance measures of team members. One pattern emerges robustly: more dispersed teams are more successful. That is distance enters positively into this regression analysis. The flip side of this prediction is that when we include distance metrics as well as developer experience measures, the explanatory power of the distance measures needs to drop. In fact, this is a pattern we do find some evidence for. This relationship is however by no means perfect, the model is based on a measure of unobservables such that this would be okay.

The remainder of this paper is structured as follows. Following a short discussion of related literature in Section 2, we explain the empirical context and provide descriptives of the data in Section 3. Section 4 outlines the model and derives empirical predictions, 5 and 6.2 tests and quantifies these predictions and finally 8 concludes.

## 2 Related literature

Our study relates to three strands of the literature: team formation and production in teams, gravity frictions in international trade – more specifically the extensive margin – and generally to the literature on the economics of OSS.

First, this work relates to the broader trade literature which makes the role of geographical distance specific in their models. Those models are typically models on network formation and accordingly, address the extensive margin of trade. The extensive margin of trade is interpreted as the outcome of a network formation model.

Chaney (2014) proposes a dynamic model to study the formation of a social network of importers and exporters. Firms may either meet at random or through the already existing network. When using the existing already existing network, firms use their already established foreign contacts to meet close - that is geographically close - firms. The model is estimated and supported by data using French exports in the years 1986 to 1992.

Another approach to network formation is proposed by Bernard et al. (2019). Using data for detailed Japanese Buyer-Supplier relationships, they argue that geographical distance plays an important role in shaping this network. To further validate their findings they use the opening of a passenger-only train line which allows for easier social network formation

while leaving costs for shipment untouched. The results support the idea that search costs are significant for describing in this case a domestic trade network.

Again, using Japanese data Bernard et al. (2020) now explores the role of social networks for the formation of research teams that are working on patents. Using the building of bridges to lower costs of meeting and face-to-face contact. The findings suggest that people who were strongly affected by this infrastructure improvement had the biggest effects on the production of knowledge.

Our work is related to understanding the geography of services. Head et al. (2009) estimate gravity equations for services trade

Within services, The development of OSS is a knowledge-intensive task of typically highly skilled individuals. Thus, our work also relates to the work on geographical aspects of knowledge production as is common in the study of academic research teams or inventor groups working on patents. Lychagin et al. (2016) analyze US firm productivity spillovers and R&D activities, and find that the exact geographic location of a firm's researchers matters for knowledge spillovers. The aspect we focus on is the citations of research papers or patents.

Atkin et al. (2022) study the effect of face-to-face interactions on patent citations between firms in Silicon Valley. Their results show that these types of interactions are an important source of agglomeration gains as they lead to knowledge flows between establishments. These knowledge flows are measured by the observed patenting behavior of these firms.

Another related field is academia. Head et al. (2019) show that distance also matters here. However, they also find that ties that bind - past co-location, co-authorship, university links - strongly reduce the distance friction suggesting a big informational friction channel. The results are driven strongly by lower quality research papers, i.e. those for which it is harder to hear randomly about. Taste heterogeneity will also play a role in shaping geography and trade. Blum and Goldfarb (2006) studying the role of distance in website traffic argue that distance effects typically occur for taste-dependent goods.

One of the closest papers to ours in terms of the role of distance in collaboration is Catalini et al. (2020), who study co-authored chemistry research papers. They find that a drop in travel costs (entry of a low-cost airline) increased collaboration greatly. In their model and data, lower travel cost is beneficial mostly for researchers with relatively weaker local peers and higher quality projects.

In terms of collaboration and diversity, a paper closely related to us AlShebli et al. (2018) who study the success of academic papers as a function of *ethnic* diversity of authors, and find a positive correlation. At the same time negative correlation between diversity and intensity of collaboration is found by Békés and Ottaviano (forthcoming) in a very different setting: professional soccer.

The role of face to face communication is studied in Battiston et al. (2021) who model communication as the tool to transmit information that helps peers (co-workers in a company) do their job better. In their model, workers trade the cost of communication to gains from it, and the benefit is more than proportional in nearby workers.

More recently, a surge of papers is using OSS data and content with related questions. For example Wachs et al. (2022) show that OSS contributors - even though most work can be conducted entirely remote - often cluster in locations like Berlin, New York, or Tokyo. The idea of clusters is also explored by Abou El-Komboz et al. (2022) who study the effect of cluster on single developers productivity. They propose to use the amount of commits of a developer as a measure of productivity and find positive effects for the effects of clusters on productivity. Moving beyond clusters and appreciating the whole distance distribution is a stream of the literature that looks at gravity-type regressions. For instance, Laurentsyeva (2019) argues that distance in different non-parametric specifications negatively affects collaboration. Common language on the other hand is important for collaboration. Using also a gravity-type regression Goldbeck (2022) uses a sample for developers living in the United States and shows that co-located developers work more intensely together. In this vain another branch of literature aims to quantify the value of OSS as for example Hoffmann et al. (2024) do. Chelkowski et al. (2021) talk about the role of large organizations after analyzing activity in 1300 projects for 21 years. They also argue for an important role in fostering collaboration. They argue that as projects mature, organizations as intermediaries are less needed.

### 3 Empirical context and data

#### 3.1 Empirical context

Open Source Software (OSS) has emerged as a foundational element of the modern digital infrastructure, playing a crucial role in almost every aspect of technology today. From operating systems like Linux to programming languages such as Python, OSS projects power a vast array of software systems and applications used globally. The collaborative and open nature of OSS enables developers from all over the world to contribute, innovate, and improve software continuously, breaking down traditional barriers associated with proprietary software development.

GitHub, the world's largest source-code management platform with about 80% of developers using it<sup>1</sup>. It is also the most widely used platform for open-source collaboration, serving as the central hub for OSS development. Founded in 2008, GitHub has grown exponentially to become the dominant platform for sharing and developing sour, with

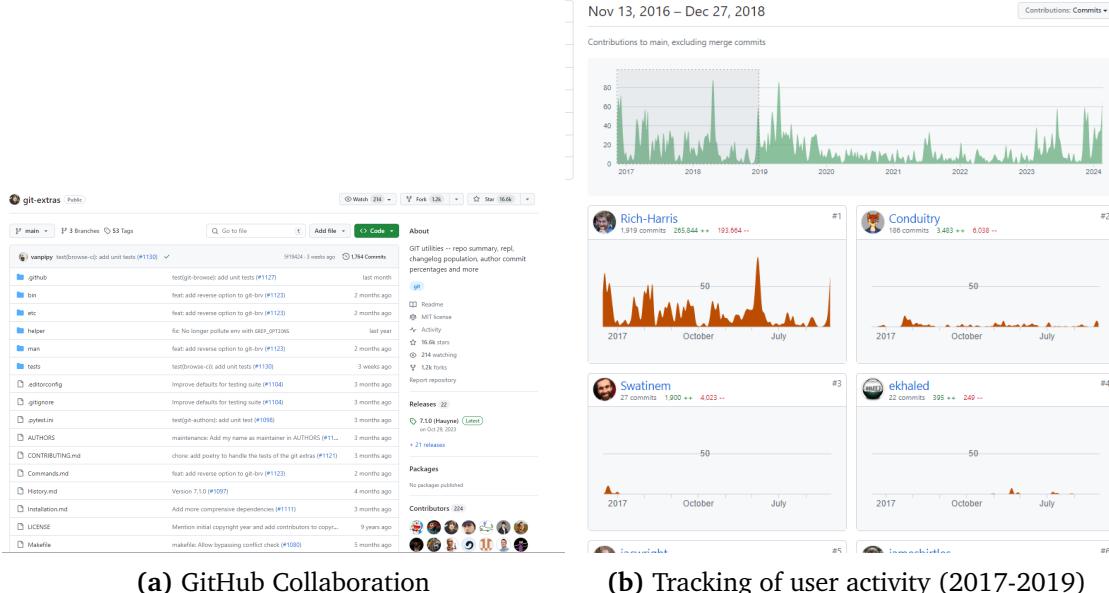
---

<sup>1</sup>According to a 2020 survey, 82% of developers use Github as collaboration tool, with GitLab, another similar platform having by 37% share, see <https://survey.stackoverflow.co/2020#technology-collaboration-tools-all-respondents>.

over 100 million registered users and more than 150 million repositories as of 2024. This platform provides a wide range of tools and services that facilitate collaboration, including version control, issue tracking, code review, and community engagement.

GitHub's impact extends beyond mere code hosting; it has also evolved into a social network where developers can share information about themselves, build professional profiles, and connect with peers globally. The platform's extensive community features, such as repositories, forks, pull requests, and "Stars" (similar to "likes" on social media), encourage collaboration and knowledge exchange on an unprecedented scale. GitHub's public repositories serve as a rich source of data for analyzing software development practices, team dynamics, and the impact of geographical dispersion on collaboration.

Given GitHub's prominence in the OSS ecosystem, it provides an ideal empirical context for studying dispersed team formation and collaboration. Unlike traditional software development environments, where teams are often co-located within the same organization, OSS projects GitHub can be developed by geographically dispersed teams. This makes it possible to explore how distance and diversity influence collaboration and project outcomes, providing insights that can be generalized to other knowledge-intensive activities.



**Figure 1: Collaboration is done mostly online**

*Notes:* Two screenshots of GitHub repositories. Panel (a) shows a repository with files, repository-level information, and a list of contributors. Panel (b) presents contributor-level activity graphs based on the number of commits per user in a given repository.

The screenshots in 1a highlight the detailed nature of the available data. Most importantly, it is possible to see which of the developers contributes in which intensity. This is something that remains in most cases impossible in other applications like patenting or academia.

### 3.2 Data overview

Our dataset comes largely from GHtorrent (see Gousios and Spinellis (2012)) and Libraries.io.<sup>2</sup>. GHtorrent provides detailed information on open-source software (OSS) projects hosted on GitHub, while Libraries.io maps software dependencies and input-output linkages among OSS projects. We integrate these input-output linkages from Libraries.io with the GHtorrent data to create our preferred measure of project success. To validate the country-level distributions derived from our user micro-data, we also rely on data from the GitHub Innovation Graph (<https://innovationgraph.github.com>).

The dataset includes software projects developed on GitHub.com between 2012 and June 2019. Initially, it contains 73,228,443 individual software projects. These projects range from simple scripts used for tasks like running linear regressions or building basic websites to advanced tools such as state-of-the-art machine learning applications. Across these projects, we observe 1,368,235,072 commits, where a commit represents a code contribution—developers write and submit code to a repository, which we use as a measure of effort.

The dataset contains 18,763,332 unique developer IDs associated with these commits. However, as is common in social network data, many projects are either not actively maintained or involve minimal contributions from developers. Our research focuses on smaller, potentially geographically dispersed teams that exhibit a certain level of activity. Therefore, we apply sample restrictions, retaining only projects with at least 4 development days and a minimum of 10 commits. This filtering reduces the sample to 10,173,295 projects.

To address the issue of bot activity on GitHub, we manually identify and exclude common bot logins. Additionally, we conservatively exclude developers who are active in more than 1,500 projects to mitigate potential biases from super high activity accounts. After these exclusions, the remaining projects are developed by 4,355,207 developers. Of these, we are able to identify the locations of 1,362,106 developers, assigning them to cities worldwide via an API.

**Repositories and Developers** Focusing specifically on projects with two developers, of which there are 1,732,734 in our dataset, we can identify the location combination for both developers in 358,748 projects, representing 20% of this subset. Developers may also belong to GitHub organizations, which can represent groups with similar interests or firms. In the subset of projects with two developers, only 7% of developers are in the same organization, and this percentage decreases to 4% in projects where we observe the locations of both developers.

---

<sup>2</sup>Libraries.io, last accessed on 07/07/2023

For each project, we record the most commonly used programming language, measured by the amount of code (in bytes) contributed in that language. This results in a total of 356 different programming languages being represented in our dataset.<sup>3</sup>

The main regressions we report will use different outcome measures to assess project success. One key measure is the number of “Stars” a project gained in the year following its initial publication. Stars on GitHub function similarly to “likes” on other social networks, indicating user appreciation of a project. An alternative measure of project quality relies on downstream dependencies. A dependency refers to the use of a project by another open-source project. Given its direct interpretation as usage, this is our preferred measure of project quality. However, it is only available for projects that are included in our second data source, Libraries.io, and where successful matching with GHtorrent data is possible.

With this description, we can dive into some more descriptive figures surrounding our data. To ensure consistence in our descriptive statistics and empirical analysis, we will only work with the previously described data including its sample restrictions.

Starting with developers, we aggregated data at the repository (project level). As Table 1 suggests 72% of projects have only a single developer. After this the amount of developers is steeply declining.

**Table 1:** How many coders develop a package?

Developer Count	Frequency	Relative Frequency
1	7,363,371	0.72
2	1,732,734	0.17
3	630,856	0.06
4	304,326	0.03
5	142,008	0.01

*Notes:* Based on the full sample of repositories, the number of developers per repository which we identify in the GHtorrent data.

Next we look at how many developers contribute to multiple projects in our dataset. We find that roughly half of developers only have one single project that fulfills the sampling criteria.

---

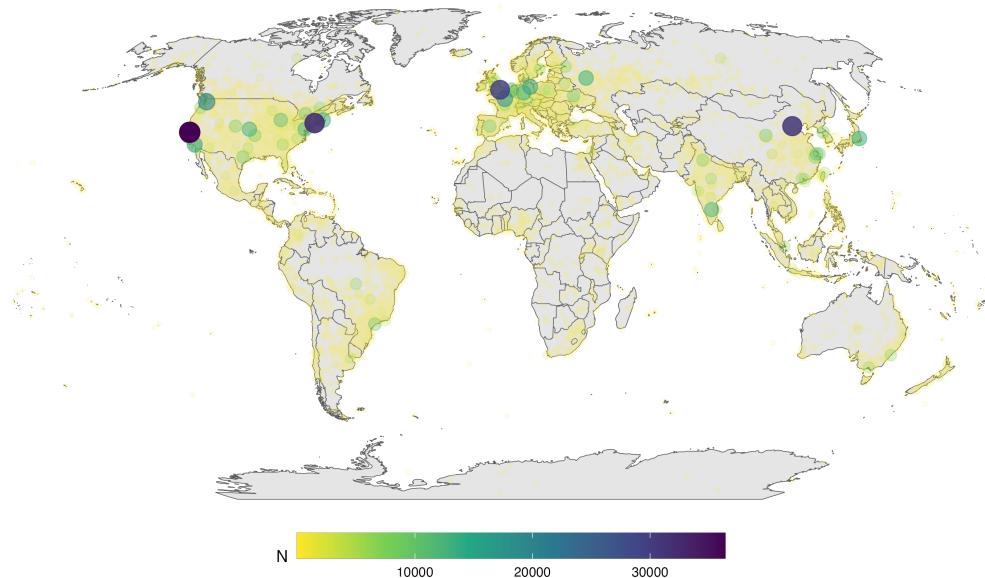
<sup>3</sup>We are currently working on grouping these languages into broader technology categories to reduce noise, as some languages are associated with only a small number of projects.

**Table 2:** How many repos developers are involved in?

	Amount projects	Frequency	Relative Frequency
1	1	2131351	0.49
3	2-5	1627010	0.37
4	6-15	460154	0.11
2	15+	136671	0.03

*Notes:* Amount of projects each developer contributes to. Consider only projects with at least 4 development days and at least 10 commits.

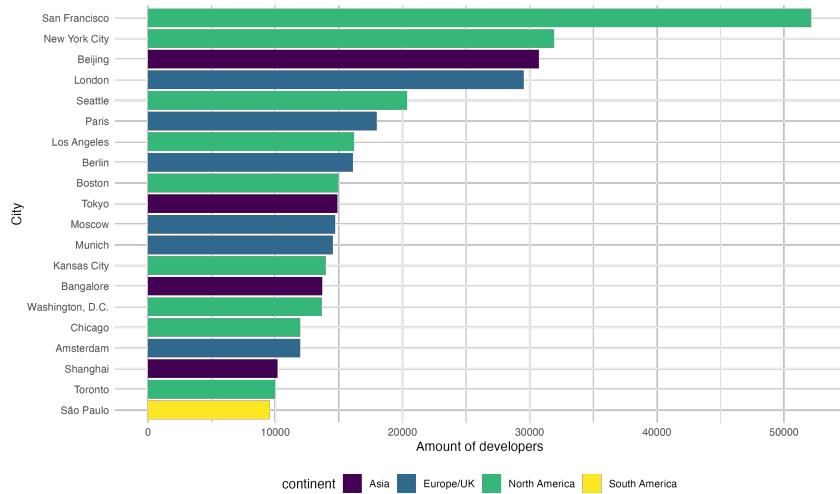
**Geography** The geographical composition is then shown in 2. This map shows that the activity is concentrated in North America, Europe, India, China, Japan and to some extent Brazil.



**Figure 2:** Developers per tenth of a square degree

*Notes:* World map depicting identified locations of open-source developers. Colour coding indicates clustering intensity with at most 5000 developers identified in San Francisco. Other technological hubs like New York, Beijing, London are clearly visible.

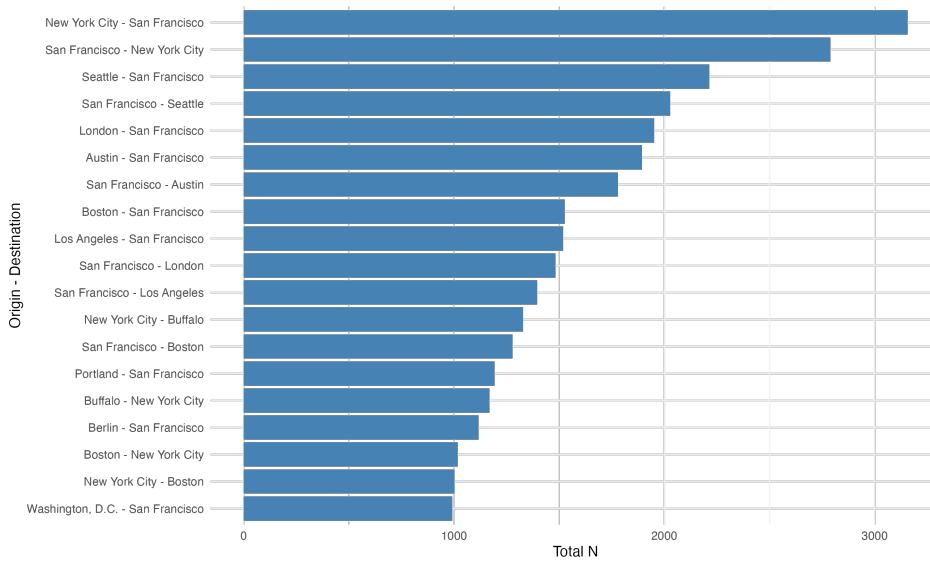
The chart in Figure 3 represents the locations of developers in the top 20 cities in our sample, the color coding represents different geographic regions. Cities in North America lead in terms of size and presence in top 20. They are followed by European centers like London and Berlin with Eastern European cities like Moscow and Kyiv also making it. Asian centers play a comparable role with Beijing and Tokyo coming on top.



**Figure 3: Contributors for Top 20 Cities**

*Notes:* Barplot illustrating the amount of open-source developers in the top 20 cities with the most developers. Colour coding based on continents.

Figure 4 depicts a horizontal bar chart that illustrates the number of shared projects between developers residing in different city pairs in 2019. The chart nicely highlights that San Francisco is the central node in open-source collaboration. Generally, all pairs include North American cities again highlighting the outsized role of North America in the context of OSS.

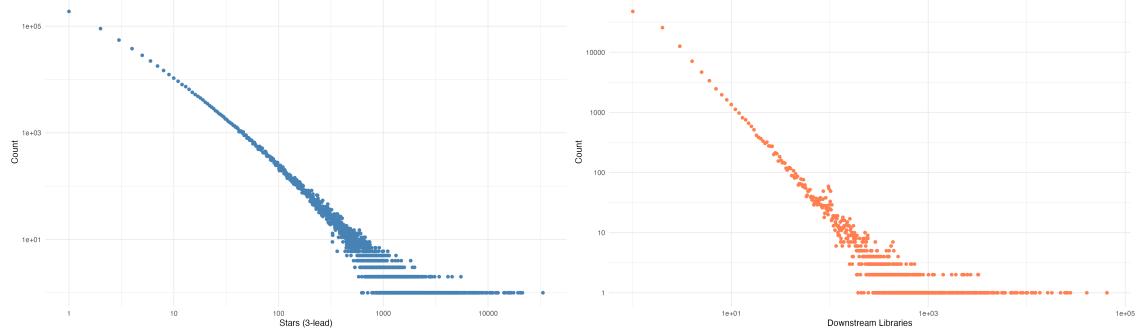


**Figure 4: Links between city pairs**

*Notes:* Barplot illustrating links between cities as measured by being active in the same repository. Same-city links are excluded from this figure. Color coding indicates world region pairs. Only one of the highest linked city pairs does not include the North American region.

The success of an open-source package is not trivial to measure. We propose two separate

measures. One makes use of the social network structure of OSS development and relies on "Stars" which users can give to other projects. The other is downstream dependencies which can be through of direct usage of projects in other software projects. Similar to Input Output linkages. Distributions are depicted in 5.



**Figure 5:** Distribution of quality measures.

*Notes:* Distribution of quality on a log scale. On the left panel the amount of stars a project received after three months. On the right panel the amount of downstream dependencies. Both exhibit a clear power law like curve. The amount of observations for the downstream libraries is sizeably lower because not all projects are hosted on package managers and can thus generate downstream linkages.

### 3.3 Work data

For all the empirical work, we restricted the dataset to study projects with up to 5 team members. Membership is defined as being an active developer: those committed in the quarter of starting the project and/or in the next one. The first committer is typically the one who hosts the project on their github page. The vast majority is single developer projects.

**Table 3:** Number of repositories by active team members

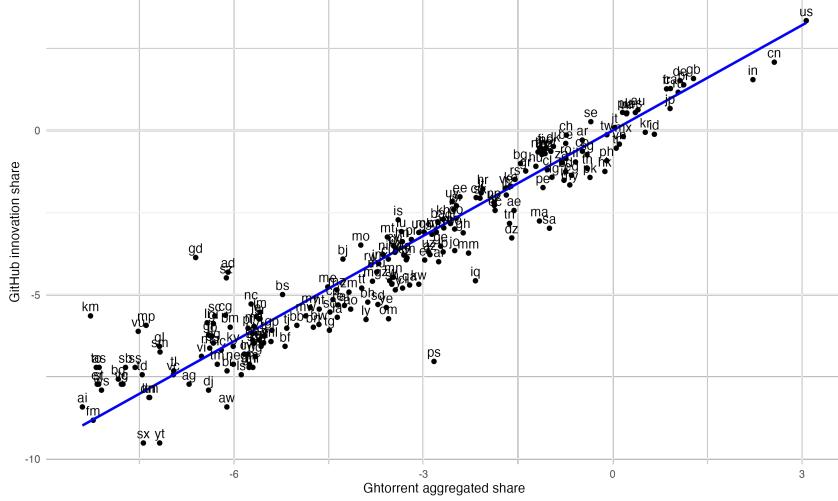
dev count	N_repos	share_repos	on_libraries_io	share_on_libraries_io
1	7,372,902	75.92%	269,218	3.65%
2	1,551,435	15.98%	50,526	3.26%
3	502,662	5.18%	12,915	2.57%
4	241,130	2.48%	4,019	1.67%
5	43,463	0.45%	437	1.01%

*Notes:* Active developers per repo, based on two quarters, up to 5 developers. Downstream dependency information come from libraries.io.

While as all repos may get stars, only a small fraction of repos are hosted on service like CRAN.

### 3.4 Data quality

We conclude this data section with a note on data quality. The location data that we exploit in this project relies on self-reported user data. Naturally, this raises the question of representatives of the data. To test this, we aggregate our data to the country level such that we have the amount of developer per country. From this, we compute the country-level shares. This we compare with the data provided from the GitHub innovation Graph which does not rely on self-reproted data but can use IP-addresses. Simply regressing these shares on each other gives us a reassuring slope of 1 with and  $R^2$  of about 0.9. The figure is presented in 6.



**Figure 6:** Comparison shares in Ghtorrent and GitHub Innovation

*Notes:* Comparison of aggregated country shares from our micro-developer level dataset and aggregated shares provided by GitHub innovation graph. Our data relies on user-self reported data wheras GitHub innovation graph uses IP addresses. The slope of log-log regressions or simple OLS on shares is at 1 suggesting our data is indeed representative.

## 4 Model of global team formation and success

### 4.1 Model

We present a model of team formation and collaboration in open-source software development to guide our empirical analysis. The main idea is that team forms to jointly produce a knowledge good of a certain quality. Quality depends on the composition of the team and on collaboration.

There are two frictions, one limiting collaboration, and one limiting team formation. Collaboration is limited by communication costs, such as language and cultural differences and time zone gaps. We will model this as an "iceberg" trade cost of ideas within the team. Team formation is further limited by incomplete sharing of credit between teammates. For example, if the main benefit of working on an open-source project is to improve job

prospects in the U.S., then developers outside the U.S. will enjoy this benefit less. Together with these frictions, the model aims to capture key features of the open-source industry:

1. Developers have heterogeneous skills that are only partially observable.
2. Collaboration across locations is costly.
3. Software quality depends on the best idea from the team.
4. There are convex returns to software quality in terms of adoption and recognition.

Figure 7 displays the outline of the model. Developer 1 and 2 are jointly considering to form a team and produce a piece of software together. The joint quality of the developers will determine the quality of the software. This, in turn, leads to “fame,” or informal recognition by the community. How much this informal recognition matters to developers determines the payoffs. The two developers may care about kudos to different degrees. This closes the model as, in equilibrium, only developers who find it worthwhile to collaborate will form a team. Others will enjoy their outside option.

**Setup** Consider a set of developers  $i = 1, \dots, N$  located in different cities. Each developer  $i$  has a skill level  $Z_i$ , which is drawn from a Fréchet distribution:

$$\Pr(Z_i \leq x) = e^{-T_i x^{-\theta}}, \quad (1)$$

where  $T_i > 0$  is an observable skill shifter and  $\theta > 0$  is a shape parameter. A higher  $T_i$  indicates higher average skills, while  $\theta$  governs the dispersion of skills. The Fréchet distribution is chosen for its tractability and its property of yielding closed-form expressions for maxima, which is crucial for our model of team production.

**Team Formation** Developers can choose to work on their own projects or join existing projects. When considering joining project  $p$ , developer  $i$  faces two key frictions:

1. A communication cost  $\tau_{ip} \geq 1$  that reduces the effectiveness of conveying ideas.
2. A participation cost  $d_{ip} \geq 1$  that captures the difficulty of joining and contributing to the project.

Both costs increase with geographic and cultural distance between the developer and the project team. We model these costs as:

$$\tau_{ip} = \text{distance}_{ip}^{\gamma_k} \quad (2)$$

$$d_{ip} = \text{distance}_{ip}^{\gamma_s} \quad (3)$$

where  $\gamma_k$  and  $\gamma_s$  are elasticities that may differ, capturing the idea that knowledge sharing and participation may have different sensitivity to distance.

**Software Production** If developer  $i$  joins project  $p$ , the quality of the resulting software depends on the best idea from the team:

$$X_p = \max_{j \in p} \{Z_j / \tau_{jp}\} \quad (4)$$

This captures the notion that in software development, a single brilliant idea or solution can drive the overall quality of the project. Given the properties of the Fréchet distribution, the quality of software  $X_p$  will also follow a Fréchet distribution with scale parameter:

$$\Phi_p = \sum_{j \in p} \tau_{jp}^{-\theta} T_{jp} \quad (5)$$

and shape parameter  $\theta$ .

**Payoffs** The payoff from a project takes the form of recognition or "fame," which increases exponentially with software quality:

$$f(X_p) = e^{\alpha_p X_p}, \quad (6)$$

where  $\alpha_p$  is a project-specific fame shifter. This fame shifter can also be interpreted as an idea quality draw that comes from some exogenous distribution. Crucially, this fame accrues only to the developer with the best idea on the team. The exponential form captures the idea that there are increasing returns to quality in terms of recognition and adoption in the open-source community. Better projects not only serve existing customers better but can also reach more customers.

**Developer's Decision** A developer  $i$  will join project  $p$  if the expected payoff exceeds their outside option of working alone. The probability that developer  $i$ 's idea is the best in the team, given their skill level  $Z_i$ , is:

$$\pi_{ip} = e^{-\Phi_{p-i} \tau_{ip}^\theta Z_i^{-\theta}} \quad (7)$$

where  $\Phi_{p-i} = \sum_{j \in p, j \neq i} \tau_{jp}^{-\theta} T_{jp}$  captures the competitiveness of the existing team. The expected payoff from joining the project is then:

$$\mathcal{U}_{ip} = \pi_{ip} e^{\alpha_p Z_i} = e^{-\Phi_{p-i} \tau_{ip}^\theta Z_i^{-\theta} + \alpha_p Z_i} \quad (8)$$

The developer will join the project if:

$$\mathcal{U}_{ip}^{1/d_{ip}} > e^{\xi Z_i} \quad (9)$$

where  $\xi$  represents the quality of solo project ideas, and the left-hand side is adjusted by the participation cost  $d_{ip}$ .

**Equilibrium** In equilibrium, the distribution of skills for developers who join project  $p$  is Fréchet with scale parameter:

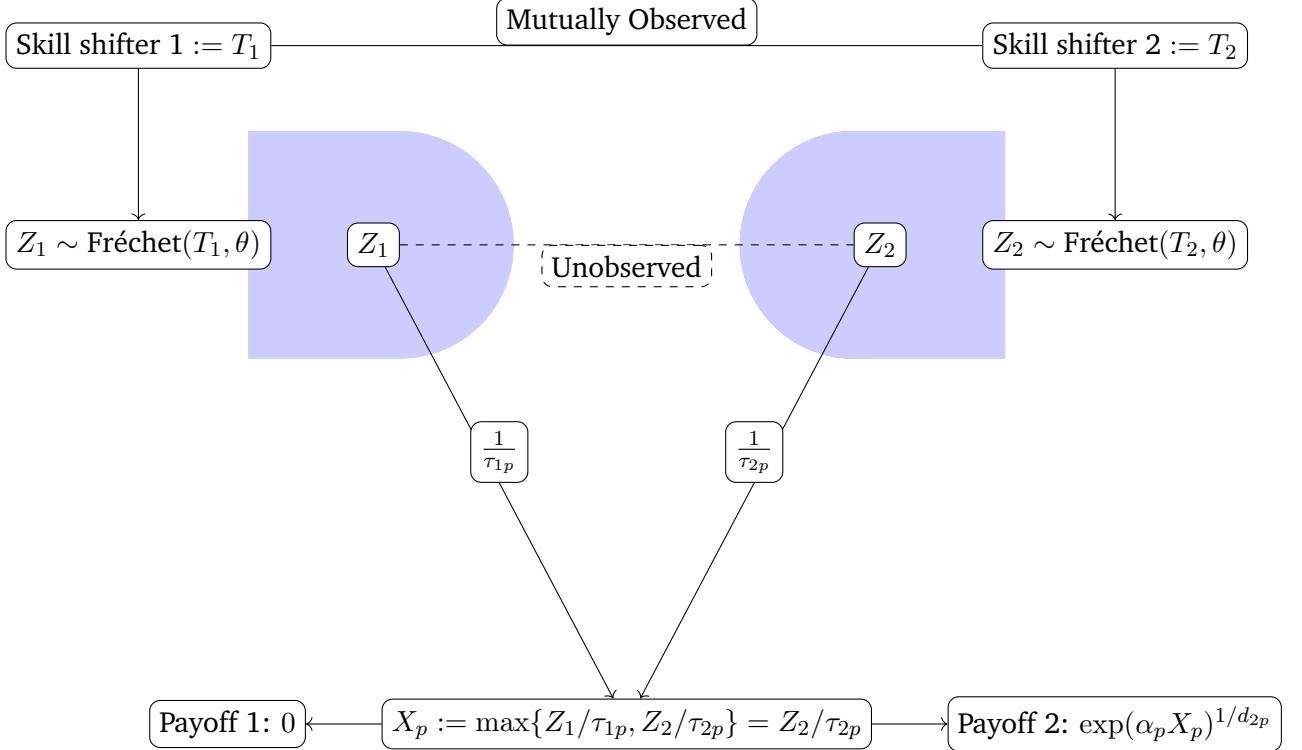
$$T_{ip} = T_i + \kappa \frac{\tau_{ip}}{d_{ip}} \sum_{j \in p, j \neq i} \tau_{jp}^{-\theta} T_{jp} \quad (10)$$

where  $\kappa$  is a constant related to the outside option. This equation captures how the selectivity of joining a project depends on both individual skills and team composition. The equilibrium is a fixed point of best responses to team composition: team members understand that others also have to agree to work on the project.

For a two-member team, this system of equations can be solved explicitly:

$$T_{1p} = T_1 + \lambda T_{2p}, \quad T_{2p} = T_2 + \lambda T_{1p} \quad (11)$$

where  $\lambda = \kappa \tau^{1-\theta}/d$  captures the strength of complementarities in the team.



**Figure 7:** Visual representation of the model.

Notes: Visual representation of the model conditional on both developers have decided to collaborate.

Developer 2 is assumed to have the higher draw from Fr\'echet distribution which leads to Developer 2 claiming the payoff for the project. Equilibrium conditions as are based on the mutually observable skills. The  $\alpha_p$  acts as an attractor to projects.

**Relaxing the payoff function** To have a more general result, we can relax the strictness of the payoff function.

[TO BE ADDED]

## 4.2 Testable Implications

This model yields several testable implications:

1. Developers are less likely to collaborate across greater distances due to higher  $\tau_{ip}$  and  $d_{ip}$ . This follows directly from equations (2) and (3).
2. Conditional on collaboration, more distant team members tend to have higher skills due to selection. This is a result of the equilibrium condition in equation (10), where higher  $\tau_{ip}$  and  $d_{ip}$  require higher  $T_i$  for participation.
3. Projects with geographically diverse teams tend to produce higher quality software, as measured by adoption or recognition. This follows from equation (5), where a diverse team allows for drawing from a larger pool of potential best ideas.
4. The effect of geographic diversity on project quality is stronger for more complex

projects where communication and idea sharing are more important. This can be captured by allowing  $\theta$  to vary with project complexity, with lower  $\theta$  (higher dispersion) for more complex projects.

5. There is a non-linear relationship between team size and project quality. While larger teams increase  $\Phi_p$  in equation (5), they also increase the selectivity of joining through equation (10).

In the following sections, we test these implications using data on open-source software development from GitHub and other sources. Our empirical strategy will involve estimating gravity-like equations for collaboration probabilities, as well as analyzing the relationship between team composition and project success.

## 5 Team formation

To test the first prediction which also acts as an assumption, we rely on in spirit on a standard gravity approach.

### 5.1 Setting up gravity

Given the large number of potential collaborators, the probability of relying on any single one of them is very small. Hence the number of links within any group of users will be well approximated by the Poisson distribution. In particular, we will take users in one city and users in another city, and form groups of such city-pairs. The number of links for a city pair is then approximately distributed Poisson, with a mean proportional to  $n_{ij} \exp(-\beta' \mathbf{d}_{ij})$  and  $n_{ij} \exp(-\gamma' \mathbf{d}_{ij})$  for dependencies., where  $n_{ij}$  is the number of user pairs in the city pair.

For example, if there are 5,000 developers in San Francisco, and 4,000 in Berlin, there are 20 million potential links,  $n_{SF,Berlin} = 20,000,000$ . By contrast, if Bielefeld has 200 developers,  $n_{SF,Bielefeld} = 1,000,000$ . The number of potential links serves as an *exposure* variable in the Poisson regression. City pairs with higher  $n$  are expected to have more links and also receive a larger weight in the regression. As a special case, if there are no developers in a city,  $n = 0$ , that city does not contribute to the regression analysis.

We estimate city-pair gravity equations with links being established by collaboration. Take developer  $j$  with  $j = 1, \dots, J$  who are contributing to a software package  $i$ . We consider two developers  $j$  and  $k$  to be connected by collaboration if they contribute to at least one software package together. Formally, let  $\mathbf{C}$  be a  $J \times J$  matrix. The  $jk$ th element of this matrix is 1 if there is at least one software project on which developer  $j$  and  $k$  collaborate, and 0 otherwise. We set the diagonal  $\mathbf{C}_{jj} = 0$  to rule out self-connections. Note that this matrix is symmetric.

If there are  $M$  cities, we can use an  $M \times J$  city-aggregator matrix to add up all the links of users belonging to the same city. Denote this aggregator matrix by  $\mathbf{M} = \mathbf{I}_M \otimes \mathbf{1}_{1 \times J}$ .

Our outcome variables will be the  $M \times M$  matrix

$$\mathbf{Y} = \mathbf{M}\mathbf{C}\mathbf{M}',$$

capturing the *number* of collaboration links between city pairs.

Take the element of this matrix corresponding to the origin city  $o$  and destination city  $d$ . By the Poisson approximation,

$$Y_{od} = \exp(\beta\tau_{od} + \nu_o + \nu_d) + \varepsilon_{od} \quad (12)$$

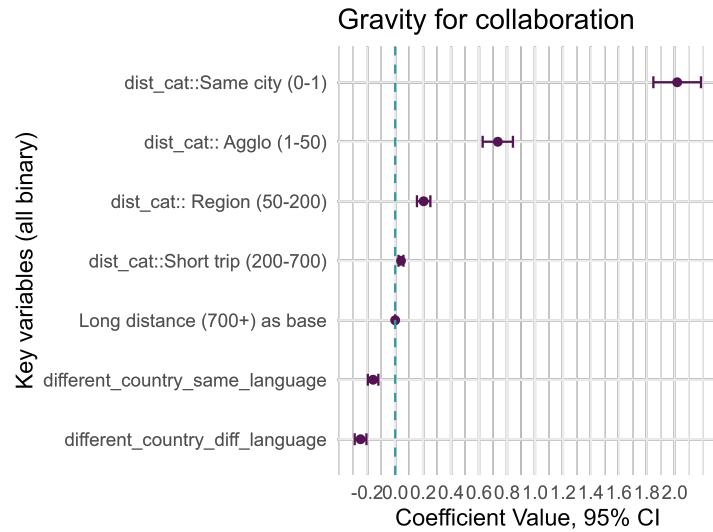
where  $Y_{od}$  are contribution links from city  $o$  to city  $d$ . Fixed effects are at the city level  $o$  and  $d$  and relevant distance metrics are captured in  $\tau_{ij}$ . Finally,  $\varepsilon_{od}$  represents an orthogonal error term. The right hand side reflects a PPML estimator ((Santos Silva and Tenreyro, 2006)) in a standard gravity framework.

The variables of interest are given by the distance vector  $\tau_{o,d}$  and its corresponding coefficients  $\beta$ . We consider standard frictions like geographic distance and common languages. To address potential non-linearities we report estimates for multiple functional forms. For more on the Poisson approximation, see Section A.1 in the Appendix.

## 5.2 P1: Nearby coders are more likely to form teams

We now estimate 12 model on the bilateral city level data. The gravity model results are shown in Table 4 while key coefficients are shown in Figure 8

**Figure 8:** Contribution and development links, core sample



*Notes:* Dependent variable is the count of links between cities. Sample: Exclude very small cities. Exclude within-organization links. PPML estimates for gravity equation

**Table 4:** Contribution and development links, core sample

Dependent Variable:	N of links between contributors		
Model:	(1)	(2)	(3)
Different city	-1.261*** (0.1055)		
ln distance   not same city	-0.0539*** (0.0060)		
dist_cat = Same city(0-1)		1.746*** (0.0772)	2.018*** (0.0858)
dist_cat = Agglomeration(1-50)		0.6351*** (0.0724)	0.7344*** (0.0873)
dist_cat = Region(50-200)		0.1905*** (0.0319)	0.2039*** (0.0307)
dist_cat = Short-trip(200-700)		0.0245* (0.0127)	0.0416*** (0.0101)
different country, same language	-0.0792*** (0.0229)	-0.1749*** (0.0215)	-0.1581*** (0.0184)
different country, diff language	-0.1910*** (0.0369)	-0.2856*** (0.0369)	-0.2476*** (0.0322)
In same organization (0-1)	5.565*** (0.0858)	5.556*** (0.0855)	
<i>Fixed-effects</i>			
city_destination	Yes	Yes	Yes
city_origin	Yes	Yes	Yes
Pseudo R <sup>2</sup>	0.84371	0.84385	0.86084
Observations	3,636,122	3,636,122	3,478,716

Notes: Dependent variable is count of links between cities. Sample: Exclude very small cities. Exclude within-organization links. PPML estimates for gravity equation. Standard errors, clustered at city\_destination & city\_origin level are in parentheses, \*\*\*: 0.01, \*\*: 0.05, \*: 0.1.

Core results (Column 3 of Table 4 and Figure 8) show strong localization and a steep decline by distance. Contributors who reside in the same city have 6.5 times ( $\exp(2.018)-1$ ) higher chance to work together than those who live in far away (700km +) segments of the same country. This suggests a massive co-location benefit. Agglomeration is also very high, suggesting a 2x higher chance. There is a sharp spatial decay and even 200km vs 700 gives only a 4% bump (and zero beyond). When physical trade cost is zero, and only face-to-face meeting costs count, the gravity model suggests massive co-location benefits, but no proximity gains beyond the commute distance. National borders also matter: Crossing a country is a 15% drop in probability, which rises to 25% if the countries don't share a language. Results are similar if we include pairs in organizations (column 2).

This confirms the role of distance. Not only is it true that more distant developers are less likely to collaborate but the point estimates are actually very large

## 6 Success and dispersion

We now bring our model predictions (2) to (5) to the data. The first prediction has been addressed in the first empirical exercise and has been motivated for the construction of the model in the first place.

### 6.1 P2 Distant team members have higher skill

The second prediction says that (conditional on collaboration), more distant team members tend to have higher skills due to selection.

Coder quality is not perfectly observed in our model. Neither it is on GitHub. On a given coder, people can look up many characteristics such as the number of repositories, past activity in terms of commits, number of stars, number of followers, personal information.<sup>4</sup>

The most relevant of these is past coding activity measured in commits. High activity is a sign of developer quality. Given the prominent display of activity on user profiles, we argue that this will capture the main mechanism discussed in the model.

We compute the average distance between the first developer and all the subsequent developers and run the following OLS regressions (6.1):

$$\bar{\text{Skill}}_{intl} = \beta_1 \text{AvgDistance}_i + \alpha_n + \lambda_t + \delta_l$$

where  $\bar{\text{Skill}}_{intl}$  can either be the average amount of commits in the previous four quarters of developers in project  $i$  or the average amount of days active in the four previous quarters,  $\beta_1$  is the parameter of interest and describes how average observed skill is related to the average distance in the project.

We first show results with two-member teams and then generalize to multiple-member teams.

Further, we include a set of fixed effects relating to the programming language, the time dimensions, the quarter, and the amount of developers in the project. Results for this regression are presented in 5.

---

<sup>4</sup>Have a look at <https://github.com/julianhinz/> to see what people looking for partners see.

**Table 5:** Dispersed teams have higher average skill

Dependent Variables:	Log(Average previous commits)	Log(Average previous days)
Model:	(1)	(2)
<i>Variables</i>		
Log(Avg Distance)	0.0366*** (0.0034)	0.0229*** (0.0023)
<i>Fixed-effects</i>		
Language	Yes	Yes
Quarter	Yes	Yes
Developer Count	Yes	Yes
<i>Fit statistics</i>		
Observations	138,423	138,423
R <sup>2</sup>	0.20480	0.16730
Within R <sup>2</sup>	0.01421	0.01148

*Clustered (Language) standard-errors in parentheses*

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

The results align with the predictions of the model given the positive elasticities. A pair of coders in a repo with a 10% higher distance between them, have on average a 0.3% higher average previous commit count.

## 6.2 P3 Dispersed teams are more successful

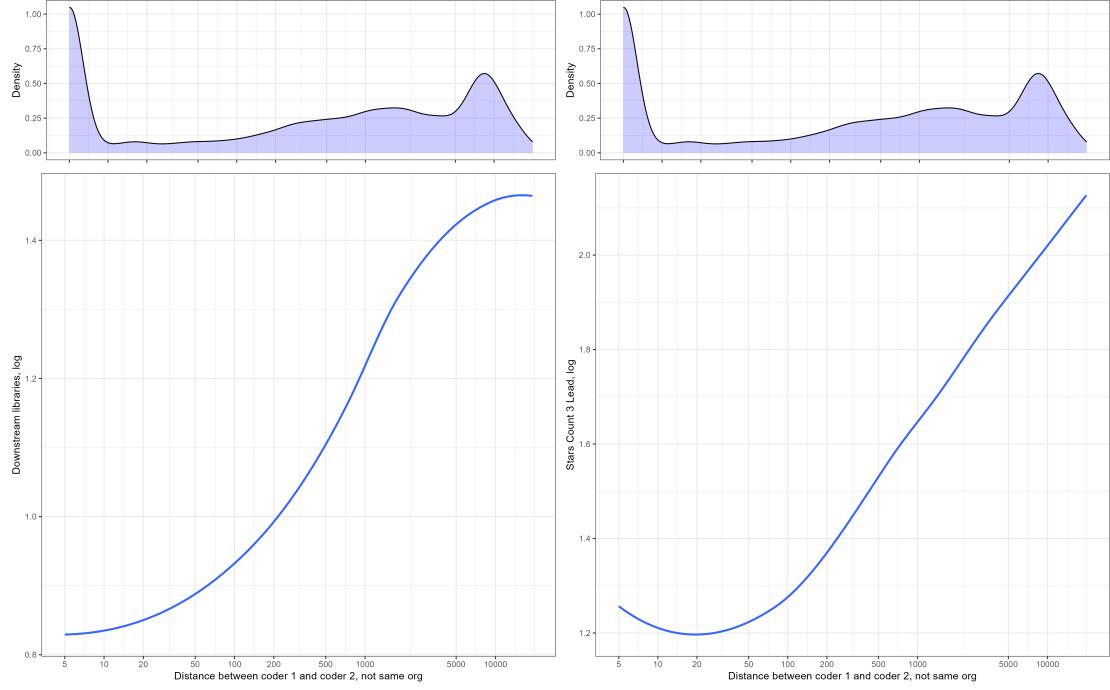
Measuring quality or success in OSS is not trivial. As outlined in the model, we think of success as the adoption of a project or visibility. Thus, to gauge the success of a project, we use stars and downstream dependencies. Downstream dependencies can be thought of as measuring technology adoption of a project. Naturally, a high amount of dependencies is a great approximation of success.

Stars, similar to likes on other platforms, can be given from users on GitHub to projects. This allows users to track the developments of the projects or simply give positive feedback. Stars will be considered with two-time horizons, once with a three-quarter lead and once with a 12-quarter lead. We aim to capture short-and long-term success of projects. We have more non-missing observations, but at the same time reverse causality is quite possible.

Before we start, please note that there is a great deal of missing location and distance information. As we saw earlier (Section ??), about 80% of location data is missing, so almost 90% of distance info is missing, too. The descriptive graph 6 however suggests that while we are working with a sample of developers, these can be regarded as representative.

Let us start by exploring two-member teams – repositories where during the *starting* quarter only two developers contributed. The graph below shows the clear log-log pattern for both downstream libraries and stars.

**Figure 9:** More dispersed teams are more successful



Notes: Repo level. Loess plot on success and distance between developer 1 and 2. Success is log number of downstream libraries (left panel) as well as log number of stars (right panel), distance is km between two developers, log scale. Kernel density by distance on upper graphs.

To get to the exact proposition, we estimate a Poisson model for the count of downstream library use  $S_{rt}$ , measured for project  $r$  at time (quarter)  $t$ , written in programming language  $l$ . The two coder qualities are denoted with  $\eta_o$  and  $\eta_d$  and measured as (ln) past number of commits:

$$S_{rt} = \exp(\beta_1 \eta_o + \beta_2 \eta_d + \beta_3 \tau_{od} + \nu_{tl}) + \varepsilon_{rt} \quad (13)$$

We control for language-time trends in popularity as it may confound the relationship.

The main results estimate regression 13 regarding propositions 3 are shown in Table 6. In column (1) we can see the coder quality, as measured with past experience, is positively correlated with success. This is true for both coders, but much more for the starting coder who can often also be thought of as the lead developer. The 0.37 log unit elasticity means that a starting coder with 10% more commits in the past 12 months, will typically start projects that can expect 4.4% more imports as a downstream repository. For the second coder it's 1.9%. Distance alone (column (2)) is positively correlated, with a point estimate of 0.147 coefficient. In column (3) we now partial out quality and see the point estimate drop to 0.104. The relationship remains once we add the complexity of the project.

Regarding Proposition (3) distance is positively correlated with success, and importantly,

when partialling out one measure of quality, the distance coefficient declines. These results confirm what we saw graphically earlier.

**Table 6:** Dispersed teams are more successful 1: downstream

Dependent Variable:	N downstream libraries			
Model:	(1)	(2)	(3)	(4)
<i>Variables</i>				
Coder 1 quality (ln past commits)	0.3684*** (0.0736)		0.3461*** (0.0696)	0.3459*** (0.0705)
Coder 2 quality (ln past commits)	0.1784*** (0.0370)		0.1604*** (0.0342)	0.1603*** (0.0343)
Distance (km, ln)		0.1475*** (0.0431)	0.1037*** (0.0399)	0.1033** (0.0401)
Repo complexity (sum commits, ln)				-0.0252 (0.1598)
<i>Fixed-effects</i>				
quarter * language	Yes	Yes	Yes	Yes
<i>Fit statistics</i>				
Observations	9,803	9,803	9,803	9,803
Pseudo R <sup>2</sup>	0.26539	0.21770	0.27416	0.27423

*Clustered (quarter-language) standard-errors in parentheses, \*\*\*: 0.01, \*\*: 0.05, \*: 0.1.*

*Poisson model, downstream library count as dependent variable.*

*Repo complexity is top total commits over the first two quarters by developers.*

*Past commits refer to the 4 quarters before the start of this repo.*

### 6.3 P4 In complex projects, distance matters more

The fourth proposition is about the complexity of the project as a mediator. We measure complexity as the sum of commits by both developers in the first two-quarters of active development. Complex projects are those with at least 90 (top 10%) total commits.

**Table 7:** Dispersed teams are more successful 2: complexity

Dependent Variable: is complex repo: Model:	N downstream libraries		
	Full sample (1)	No (2)	Yes (3)
	<i>Variables</i>		
Distance (km, ln)	0.1037*** (0.0399)	0.0860** (0.0394)	0.2485*** (0.0919)
Coder 1 quality (ln past commits)	0.3461*** (0.0696)	0.3232*** (0.0557)	0.0280 (0.0979)
Coder 2 quality (ln past commits)	0.1604*** (0.0342)	0.1561*** (0.0253)	0.0943 (0.0737)
<i>Fixed-effects</i>			
quarter * language	Yes	Yes	Yes
<i>Fit statistics</i>			
Observations	9,803	8,929	750
Pseudo R <sup>2</sup>	0.27416	0.25175	0.69514

*Clustered (quarter-language) standard-errors in parentheses, \*\*\*: 0.01, \*\*: 0.05, \*: 0.1.  
Poisson model, downstream library count as dependent variable. Complex repo is top 10% in terms of total commits over the first two quarters by developers. Past commits refer to the 4 quarters before the start of this repo.*

Indeed, we find that more complex projects – where more efforts were invested in terms of commits – show a substantially stronger distance elasticity.

The effect of geographic diversity on project quality is stronger for more complex projects where communication and idea sharing are more important.

## 7 Multi-developer teams

### 7.1 Extensions to multiple member teams

We now repeat all previous exercises for teams with up to 5 members. Bilateral distance is replaced with average distance between the first developer and others.

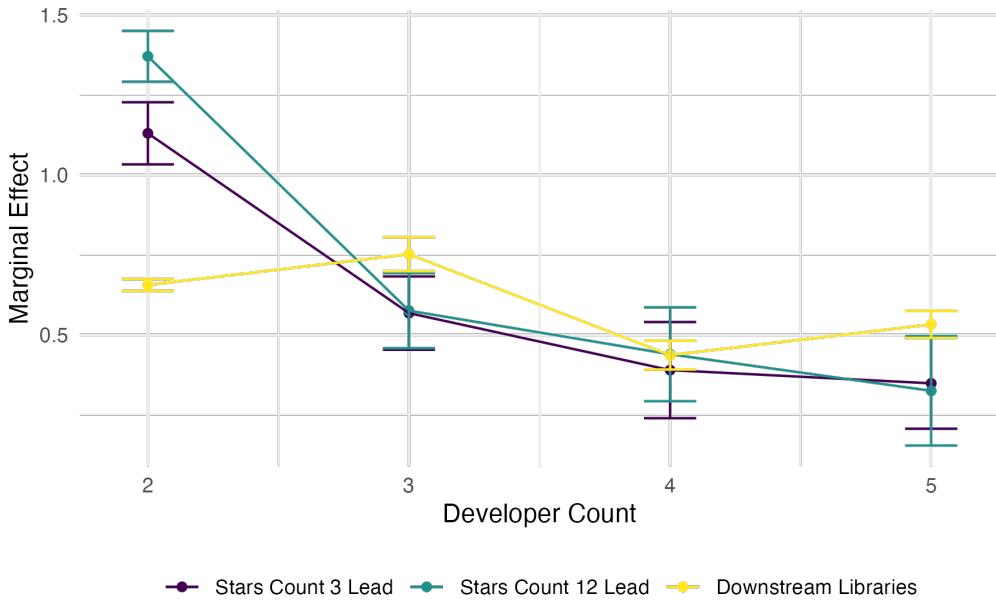
[To be added]

### 7.2 P5 Non-linearity in developer group size

The last proposition was about team size: suggesting a non-linear effect in team size growth. While the move from one to two developers is associated with big gains, adding even more developers beyond that is not as important. We estimate the following regression on our sample of projects:

$$Y_{itl} = \gamma \#Developers_i + \lambda_t + \delta_l$$

The marginal effects which refers to the differences between categories are displayed in 10.



**Figure 10:** Non-linear effect of developer count increases on different software quality metrics.

For stars, the predictions of the model align with the data. While going from one developer to two is really beneficial, adding yet another developer only leads to limited gains. For downstream dependencies however, we can not confirm the non-linear nature of a team size effect as the estimate is relatively stable throughout.

## 8 Conclusion

In several areas of the knowledge intensive economy, people form ad-hoc project teams to write music, do chemistry research or write software code. This paper studies these scenarios. Spatially scattered developers found small teams, and despite all work is done online, space emerges as an important friction in team formation. It also acts as engine of selection, yielding a positive correlation between success of teams and distance.

## References

- Allen, Treb**, “Information frictions in trade,” *Econometrica*, 2014, 82 (6), 2041–2083.
- AlShebli, Bedoor K., Talal Rahwan, and Wei Lee Woon**, “The preeminence of ethnic diversity in scientific collaboration,” *Nature communications*, 2018, 9, 1–10.
- Atkin, David, M. Keith Chen, and Anton Popov**, “The Returns to Face-to-Face Interactions: Knowledge Spillovers in Silicon Valley,” Working Paper 30147, National Bureau of Economic Research June 2022.
- Battiston, Diego, Jordi Blanes i Vidal, and Tom Kirchmaier**, “Face-to-Face Communication in Organizations,” *The Review of Economic Studies*, March 2021, 88 (2), 574–609.
- Bernard, Andrew B, Andreas Moxnes, and Yukiko U Saito**, “Production networks, geography, and firm performance,” *Journal of Political Economy*, 2019, 127 (2), 639–688.
- , — , and — , “The Geography of Knowledge Production: Connecting Islands and Ideas,” Technical Report, Working paper 2020.
- Blum, Bernardo S and Avi Goldfarb**, “Does the internet defy the law of gravity?,” *Journal of international economics*, 2006, 70 (2), 384–405.
- Békés, Gábor and Gianmarco Ottaviano**, “Cultural Homophily and Collaboration in Superstar Teams,” *Management Science*, forthcoming.
- Catalini, Christian, Christian Fons-Rosen, and Patrick Gaulé**, “How Do Travel Costs Shape Collaboration?,” *Management Science*, 2020, 66 (8), 3340–3360.
- Chaney, Thomas**, “The network structure of international trade,” *The American Economic Review*, 2014, 104 (11), 3600–3634.
- Chelkowski, T., D. Jemielniak, and Kacper Macikowski**, “Free and Open Source Software organizations: A large-scale analysis of code, comments, and commits frequency,” *PLoS ONE*, 2021, 16.
- Eaton, Jonathan, Samuel S Kortum, and Francis Kramarz**, “Firm-to-firm trade: Imports, exports, and the labor market,” Technical Report, National Bureau of Economic Research 2022.
- El-Komboz, Lena Abou, Thomas Fackler et al.**, “Productivity Spillovers among Knowledge Workers in Agglomerations: Evidence from GitHub,” in “VfS Annual Conference 2022 (Basel): Big Data in Economics” number 264083 Verein für Socialpolitik/German Economic Association 2022.

**Goldbeck, Moritz**, “Bit by bit: colocation and the death of distance in software developer networks,” Technical Report, ifo Working Paper 2022.

**Gousios, Georgios and Diomidis Spinellis**, “GHTorrent: GitHub’s data from a firehose,” in “2012 9th IEEE Working Conference on Mining Software Repositories (MSR)” IEEE 2012, pp. 12–21.

**Head, Keith, Thierry Mayer, and John Ries**, “How remote is the offshoring threat?,” *European Economic Review*, 2009, 53 (4), 429–444.

—, **Yao Amber Li, and Asier Minondo**, “Geography, Ties, and Knowledge Flows: Evidence from Citations in Mathematics,” *The Review of Economics and Statistics*, 10 2019, 101 (4), 713–727.

**Hoffmann, Manuel, Frank Nagle, and Yanuo Zhou**, “The Value of Open Source Software,” *Harvard Business School Strategy Unit Working Paper*, 2024.

**Laurentsyeva, Nadzeya**, “From friends to foes: National identity and collaboration in diverse teams,” Technical Report, CESifo Discussion Paper 2019.

**Lychagin, Sergey, Joris Pinkse, Margaret E. Slade, and John Van Reenen**, “Spillovers in Space: Does Geography Matter?,” *The Journal of Industrial Economics*, 2016, 64 (2), 295–335.

**Silva, J M Santos and Silvana Tenreyro**, “The Log of Gravity,” *Rev. Econ. Stat.*, 2006, 88 (4), 641–658.

**Wachs, Johannes, Mariusz Nitecki, William Schueler, and Axel Polleres**, “The Geography of Open Source Software: Evidence from GitHub,” *Technological Forecasting and Social Change*, 2022, 176, 121478.

## A Appendix

### A.1 Poisson approximation

We have a binary outcome  $y_i \in \{0, 1\}$  for  $i = 1, 2, \dots, N$ , with  $N$  very large. The probability of success is given by logit,

$$\Pr(y_i = 1|X_i) = \Pi(\beta X_i),$$

with  $\Pi(z) = e^z / (1 + e^z)$ .

The explanatory variable is vector valued and may take  $G$  distinct values with  $G \ll N$ . For example, it may be a same city dummy ( $G = 2$ ) or the pairwise distance between  $K$  cities ( $G = K^2 \ll N$ ).

The log likelihood contribution of observation  $i$  is

$$\ln \mathcal{L}_i = y_i \ln \Pi(\beta X_i) + (1 - y_i) \ln[1 - \Pi(\beta X_i)]$$

so that the total log likelihood in the sample is

$$\ln \mathcal{L} = \sum_{i=1}^N y_i \ln \Pi(\beta X_i) + (1 - y_i) \ln[1 - \Pi(\beta X_i)].$$

Consider a change of variables. The values in the sum will be repeated very frequently, because there are only  $G$  distinct values that  $X_i$  can take. Let  $Z_g$  denote the  $g$ th possible value of  $X_i$ . For example, for a same-city dummy,  $Z_1 = 0$  and  $Z_2 = 1$ .

Let  $n_g := \|i : X_i = Z_g\|$  denote the number of observations for whom  $X_i$  takes its  $g$ th possible value. We will also call this group  $g$ .

Within this group, some observations have  $y_i = 1$ . Their count will be  $n_{g1} := \|i : X_i = Z_g, y_i = 1\|$ . Similarly, we denote  $n_{g0} := \|i : X_i = Z_g, y_i = 0\|$ . Trivially, we have  $n_g = n_{g0} + n_{g1}$ .

For example, suppose  $X_i$  is a same-city dummy. Then  $G = 2$ , with  $g = 1$  denoting not in the same city.  $g = 2$  being the same city. Then  $n_{11}$  denotes the number of user pairs that are *not in the same city* and *have a link*.

Given this change of variables, we can rewrite the sum of log likelihood as

$$\ln \mathcal{L} = \sum_{g=1}^G n_{g1} \ln \Pi(\beta Z_g) + n_{g0} \ln[1 - \Pi(\beta Z_g)].$$

Maximizing with respect to  $\beta$  yields

$$0 = \sum_{g=1}^G n_{g1} \frac{\Pi'(\beta Z_g)}{\Pi(\beta Z_g)} Z_g - n_{g0} \frac{\Pi'(\beta Z_g)}{1 - \Pi(\beta Z_g)} Z_g = \sum_{g=1}^G \frac{\Pi'(\beta Z_g)}{\Pi(\beta Z_g)[1 - \Pi(\beta Z_g)]} Z_g [n_{1g} - \Pi(\beta Z_g) n_g].$$

The term in brackets is the error term in a regression trying to predict the number of links on group  $g$ . That error term is orthogonal to  $Z_g$ . The fraction at the beginning provide some weighting across groups,

$$\frac{\Pi'}{\Pi(1 - \Pi)} = 1 - \Pi(\beta Z_g).$$

When predicting links in a sparse graph, the probability of  $y_i = 1$  will be very small, so that  $\Pi \approx 0$  and the weight is  $\approx 1$ .

Then the first-order condition for ML can then be approximated by

$$0 = \sum_{g=1}^G Z_g [n_{1g} - e^{\beta Z_g} n_g].$$

This is *exactly* the FOC for a Poisson ML, with  $n_g$  as an exposure variable.