Success and geography: Evidence from open-source software

Gábor Békés[1] Julian Hinz[2] **Miklós Koren**[3] Aaron Lohmann[4]

April 4, 2024

[1]Central European University, HUN-REN KRTK, CEPR
[2]Bielefeld University, Kiel Institute for the World Economy
[3]Central European University, HUN-REN KRTK, CEPR, Cesifo
[4]Bielefeld University, Kiel Institute for the World Economy

University of Bologna — April 4, 2024

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.

Introduction

Big Picture:

- How and where good Open Source Software (OSS) is produced.
- How dispersed developers can create high quality software.

How and where good Open Source Software (OSS) is produced?

Interest in geography of development

- Are there spatial frictions even though all online?
  - weightless economy – no transport cost, face-to-face interaction limited, collaboration online.
- How combination of developers – in terms of location – relate to success (users)?

How and where good Open Source Software (OSS) is produced?

Interest in geography of development

- Are there spatial frictions even though all online?
  - weightless economy – no transport cost, face-to-face interaction limited, collaboration online.
- How combination of developers – in terms of location – relate to success (users)?

Data

- Writing code together – Collaboration (Github)
- Using other people's code – imported dependencies (Libraries.io).

What we do

- Compare probability of collaboration and its success as function of spatial dispersion

Open Source Software (OSS) is HUGE

- Software industry – 1% of global GDP
- 90+% of software has open source components
- `GitHub` alone hosts over 400 million repositories by $100m+$ developers
- User value estimated USD 8.8 trillion globally (Hoffmann et al., 2024)

Open Source Software (OSS) is everywhere

OSS plays an important roles in

- Websites (JavaScript)
- Operating systems (Linux, Android)
- Data (R Tidyverse, Python Pandas, Julia)
- Machine Learning and AI (PyTorch, LLaMA)

OSS mostly free, but present in fee-based platforms

- Overleaf

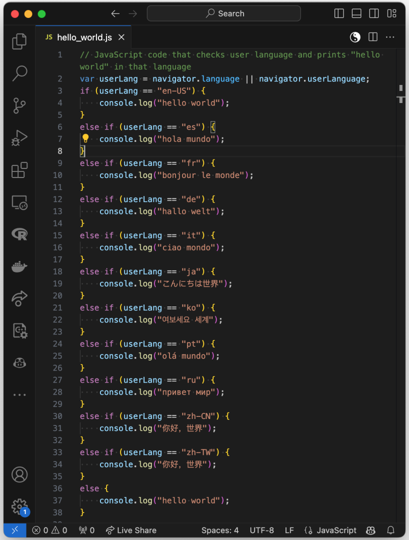
Focus on JavaScript

- `JavaScript` is one of the biggest programming languages

→ used in web development and app development

- `NPM` is a package manager

→ organizes packages and provides access



An Italian university landing page

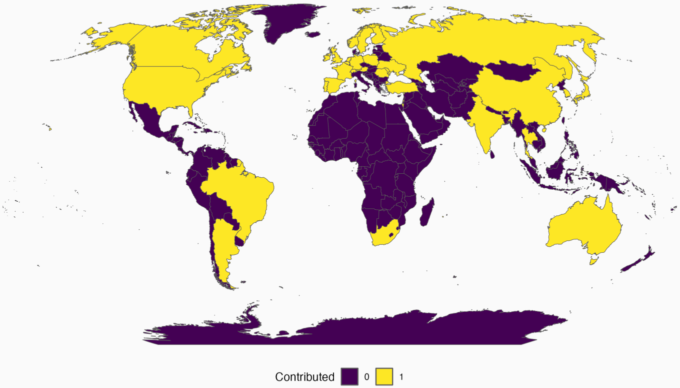
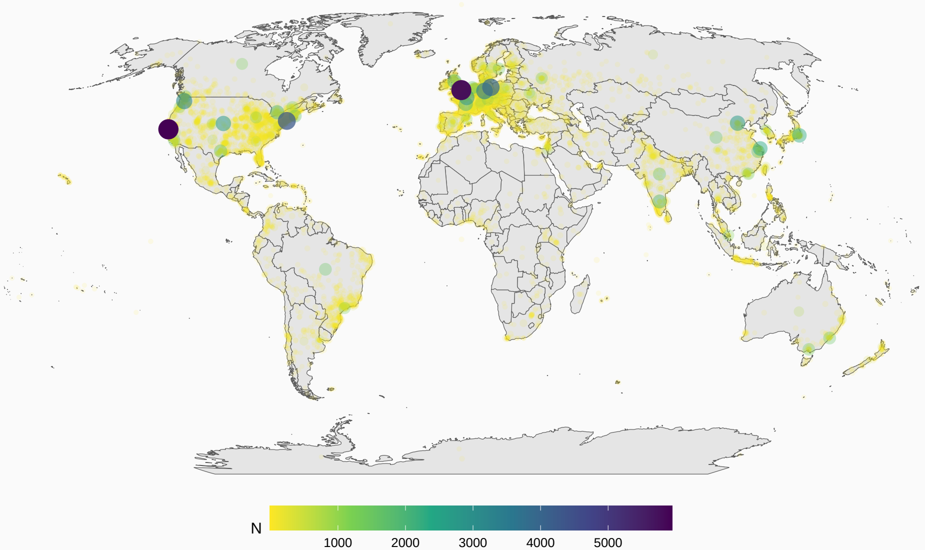
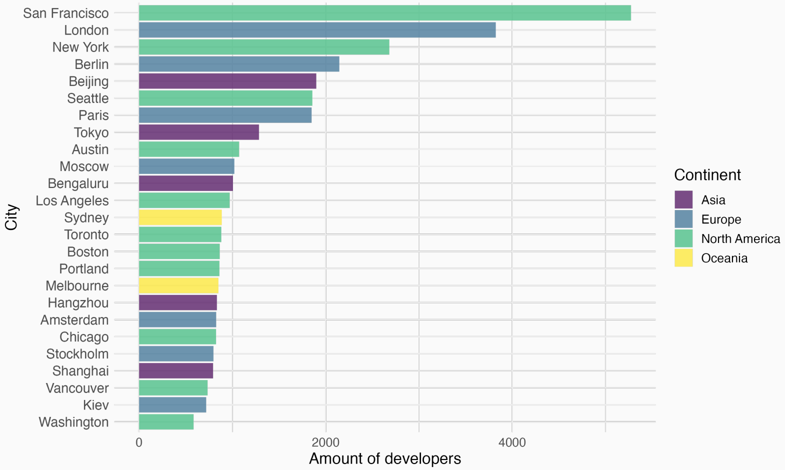


Figure 1: World countries in which at least one developer has contributed to top 3 OSS behind site: `jQuery`, `OWL.carousel` or `Modernizr` as of June 2019

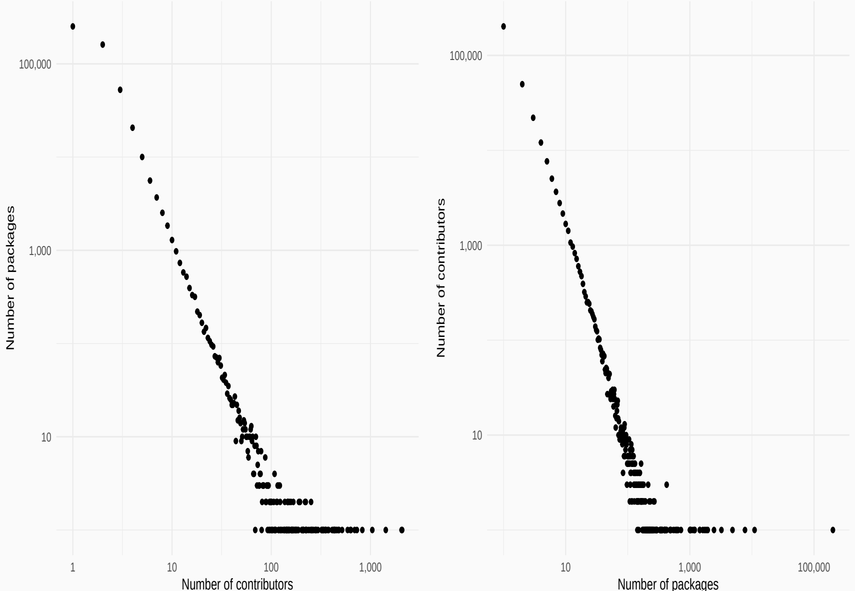
Global industry: Number of JavaScript developer per city



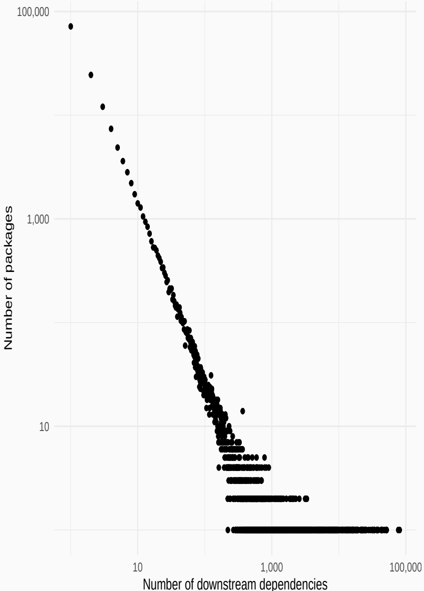
Dispersion and concentration: top cities per number of developers

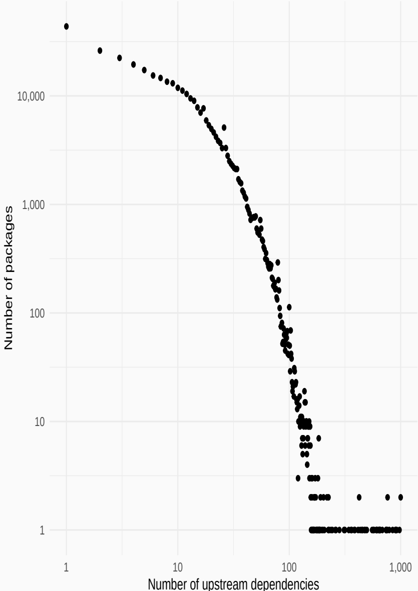


Large variation in number of projects and developers

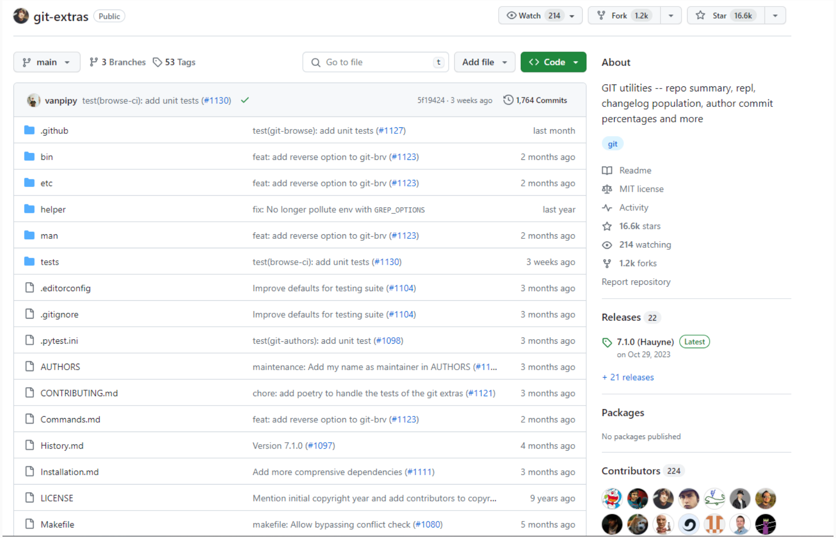


With limits on how many projects one imports

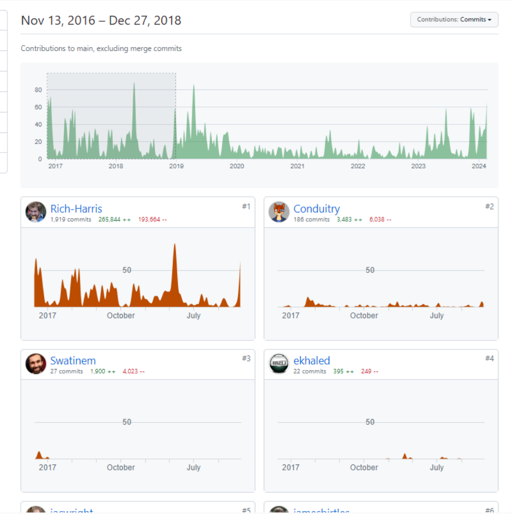




Collaboration is done mostly online



Collaboration is done mostly online

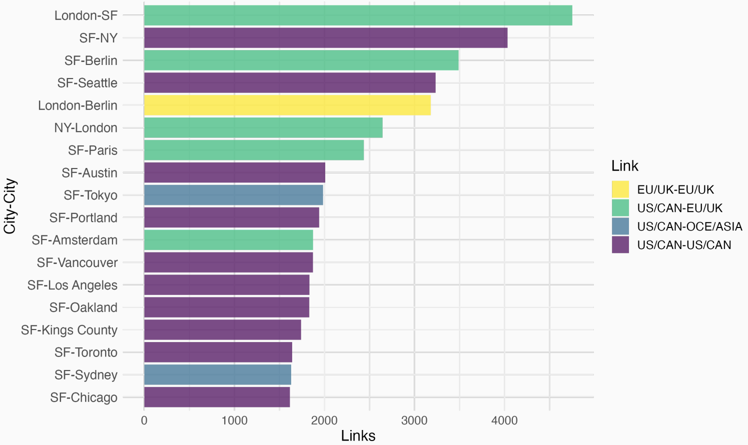


… but personal contacts still matter

- Personal meeting, esp. workplace (CEU, Oracle)
- Local community events, science parks (Xaccelerator)
- Regional events (R Ladies Auckland, VDSG Meetup, PyData Berlin)
- Conferences 1: dozens of events every month such CityJS Berlin, React Summit US,
- Conferences 2: developers directly such as Node-js fwdays23 in Kyiv, where new packages are presented.

- Learn about packages, devs: online forums, Stack Overflow, Twitter

Collaboration across cities is mostly North-North



Most frequent city-pairs for repos developed from 2 cities

Related literature

- **Geographical Distance / Network formation / Agglomeration**: Chaney (2014) Bernard et al. (2019) Davis and Dingel (2019) Bailey et al. (2021), Atkin et al. (2022)
- **Gravity: Digital:** Blum and Goldfarb (2006) Anderson et al. (2018)
- **Frictions in services:** Stein and Daude (2007) Bahar (2020)
- **Patents and science**: Bircan et al. (2021), Head et al. (2019), Jaffe et al. (1993), Singh (2008) AlShebli et al. (2018), Li (2014)
- **OSS**: Lerner and Tirole (2002) , Laurentsyeva (2019) Wachs et al. (2022) Fackler et al. (2023)

Open source software vs patents and academia

- R&D and patenting
  - Need machines, secrecy, often top-down
  - Distance matters in collaboration
  - More cited patents – geographically focused authors
- Science (math, academic papers)
  - Similar, but often longer projects, not open, F2F important to think and discuss
  - Distance matters in collaboration
  - Major role of top Universities / Centers

Today

- OSS and data
- The role of space in collaboration
    - Gravity
    - Success

Open source software data

Open Source vocabulary

- `Package`: A unit of software, provision of a (bundle of) functionality
- `Project`: A software project offering solution to a use case. Typically one package, but may be more.
- `Repository`: A storage for one project (what we observe)
- `Commit`: The smallest unit of contribution

- `Git`: Distributed version control system for software projects
- `GitHub`: A platform to collaboratively work on software projects

- `Dependency`: An imported package that provides a functionality

Data from GHTorrent and Libraries.io

Collaboration — Working on the same code with others

- `GHTorrent`: Tracks metadata on `GitHub` usage

→ Commits, locations and user organisations

- Row: One commit from a developer to a repository
- Focus on links: binary if a developer committed at all to a repository

Dependencies — Sourcing of intermediate inputs

- **`Libraries.io`**: Tracks data on single software repositories

→ dependency linkages

- Row: An imported dependency (package) to repo 1 from repo 2

→ Can be mapped to repositories on GitHub

Scope of data

- Data coverage: 2013 – 2019
- We know location as city for developers
- Contributions by 217K developers,
- 300K repos
- 17% of repos have multiple developers (ie have collaboration)
- 70K organizations, with 120K developers

Sample design: exclude later arrival, bug-fixers

We focus on collaborating partners, who are likely to have interaction, joint decisions.
Exclude

1. Bugfixers – as external ”consultants” who come in help solve a problem
   - Less than 4 commits or 1% of commits | less than 10 commits total
2. Late arrivals – developers who take over maintenance or add important extensions late
   - Developers who first commit 730 days after the first commit

As we look at dynamics, we focus on projects we see the first commit, ie after 2013.

Raw data to regressions

- Collaboration – link developers who contribute to the same repo.
- Dependencies – link developers from one package using another

- One observation is one link
- Aggregated at city (city pair) level

Collaboration

- Start with the developer's link to a repository (via commits)
- Directed but (mostly fully) symmetric
- Transform it to developer to developer links
- Aggregate at city level

Links in the contribution network

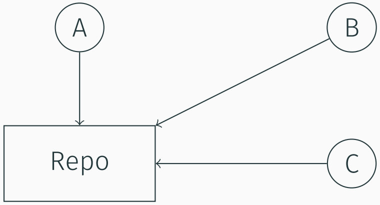


Figure 2: Developers commiting to a repository.

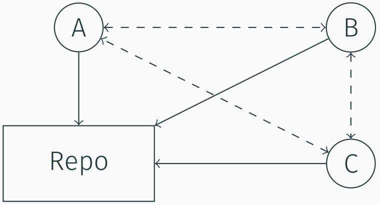


Figure 3: Developers commiting to a repository including implied contributor to contributor links.

Solid lines are what we **observe**. Dashed lines is what we **infer**.

Aggregation – weights

- In a repo, all developers create links with each other
- If two people have 3 repo together, will generate 3 links

- Also look at *intensive* margin – weighted by commits

Organisations

- Github collaboration system
- Mostly amateurs (like CEU Econ)
- Includes corporations (like Oracle)

- Today: mostly focus outside organizations

Estimating gravity

Gravity: finding a partner

- The role of distance in finding a partner
- Search and maintenance

- Each developer can choose any partner: logit
- Aggregate + transform: Poisson at city pair level: number of links as function of distance
- *(Yes, like structural gravity: PPML, FEs)*

MORE: From logit to Poisson

Gravity: finding a partner

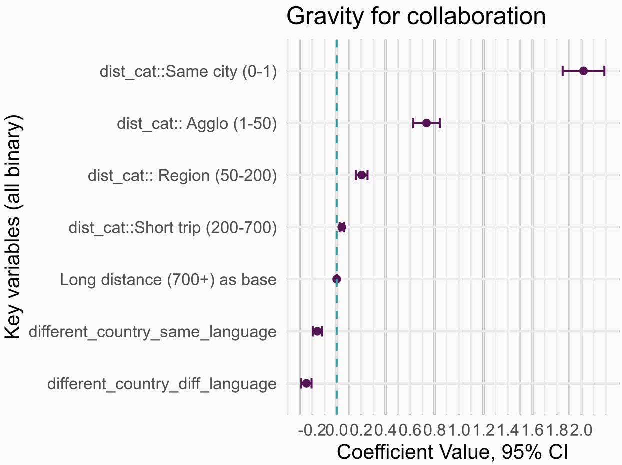
$$\Pr(Y_{od}|x_o, x_d, d_{od}) \approx \text{Poisson}[N_o \times N_d \times \exp(\beta_1 x_o + \beta_2 x_d + \beta_3 d_{od})]$$

- Outcome: Number of links between cities o, d
- d_{od} Distance measured as a set of indicators / log-linear
- Origin and destination city FE
- $N_o \times N_d$ -Exposure: Number of developers in city o x d

Modelling search and maintenance costs

- Meeting – distance in terms of travel
  - Same city – e.g. universities, office parks
  - Agglomeration (1-50km) – regional events
  - Regional (50-200km) – national conferences
  - Short trip (200-700km) – big conferences
  - Beyond 700km (*as base*) – global events
- Travel difficulty
  - Crossing borders
  - Crossing borders — different language

Results 1: More work together when closer



Gravity 1: N of links between cities declines with distance

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***	2.018***
		(0.0772)	(0.0858)
dist_cat = Agglomeration(1-50)		0.6351***	0.7344***
		(0.0724)	(0.0873)
dist_cat = Region(50-200)		0.1905***	0.2039***
		(0.0319)	(0.0307)
dist_cat = Short-trip(200-700)		0.0245*	0.0416***
		(0.0127)	(0.0101)
different country, same language	-0.0792***	-0.1749***	-0.1581***
	(0.0229)	(0.0215)	(0.0184)
different country, diff language	-0.1910***	-0.2856***	-0.2476***
	(0.0369)	(0.0369)	(0.0322)
In same organization (0-1)	5.565***	5.556***	
	(0.0858)	(0.0855)	

Results 1: Comparisons

- Math academic papers (Head et al., 2019) – similar
- Patents (Li, 2014): smaller point estimates here, esp cross-country

Results 2: Commits as kinda intensive margin

- Special feature of coding – intensive margin

- Look at commits – number of changes in code
- Bit like extensive margin

Gravity 2: Co-location = more intensive work

Dependent Variables:	N links	commit share
Model:	(1)	(2)
Variables		
dist_cat = Samecity(0-1)	2.018***	0.7564***
	(0.0858)	(0.1309)
dist_cat = Agglo(1-50)	0.7344***	0.1838
	(0.0873)	(0.1410)
dist_cat = Region(50-200)	0.2039***	0.0906
	(0.0307)	(0.0795)
dist_cat = Shorttrip(200-700)	0.0416***	-0.0192
	(0.0101)	(0.0399)
Fixed-effects		
city_destination	Yes	Yes
city_origin	Yes	Yes
Fit statistics		
Pseudo R^2	0.86084	0.52444
Observations	3,478,716	451,423

Origin, destination city FE, Clustered (city_destination & city_origin) standard-errors in parentheses

Robustness

- Maybe a few very large repositories dominate and flatten the curve. No
- Also no huge difference excluding few largest cities

Estimating success and dispersion

Success (popularity) and spatial dispersion

- Popularity = measures the number of other packages which declare a dependency on a the repository in NPM
- Measures on spatial dispersion
- Controls

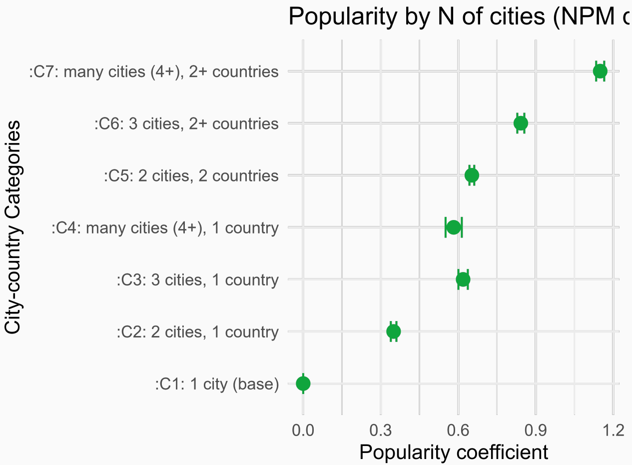
Success (popularity) and spatial dispersion

$$\Pr(Y_i|.) \approx \text{Poisson}[\exp(\beta_1 cities_i + \beta_2 countries_i) + \gamma Z]$$

- Outcome: Number of repos importing this repo i

- $countries_i$ number of countries
- $cities_i$ number of cities
- Z: f(number of developers), f(age of project)

Results 1: More popular dependency - higher spatial dispersion



Results 1: More popular dependency - higher spatial dispersion

Dependent Variable:	N Dependents (NPM)		
Model:	(1)	(2)	(3)
Count of cities	0.4075***	0.2306***	
	(0.0403)	(0.0491)	
Count of countries	0.3431***	0.3057***	
	(0.0628)	(0.0637)	
City cat × CI2 × 2cities			0.3851***
			(0.0813)
City cat × CI3 × 3cities			0.4925***
			(0.1242)
City cat × CI4 × many cities(4+)			0.6543***
			(0.1642)
Country cat × CO2 × 2 countries			0.2461***
			(0.0813)
Country cat × CO3 × many countries(3+)			0.6269***
			(0.1462)
Constant	1.745***	1.148***	1.673***
	(0.0548)	(0.0857)	(0.0674)
Age, N_Dev	No	Yes	Yes
Commits	No	No	No
Coders	No	No	No
Pseudo R^2	0.05100	0.11532	0.11586
Observations	36,491	36,491	36,491

Packages built by *more* dispersed people will be used more. Why?

1. Reverse causality: diverse developer pool – larger market reach

2. Selection I: Random / assortative matching + large cities having best developers
3. Selection II: Multiple skill-set of developers + search costs – high FC to work outside city – best developers select search more + get into good projects
4. Selection III: give high collaboration costs across cities, once started, teams work more

5. Causal I: Diversity helps via specialized knowledge across cities
6. Causal II: Diversity creates better ideas (allow skipping group-think)

1. Reverse causality?

- Is dependency import affected by geography?
- Developers from larger cities gain greater audience

Preparation: Aggregating dependencies to city level

- We observe a repository importing another one as dependency.
- Directed, not symmetric
- Transform it to developer-to-developer links
    - Use knowledge of producers of the dependency as well
- Aggregate at city level

Links in the dependency network

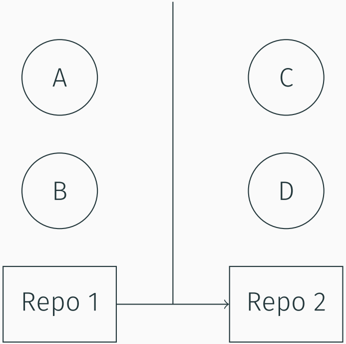


Figure 4: Dependency of repository 1 on repository 2 with the respective developers.

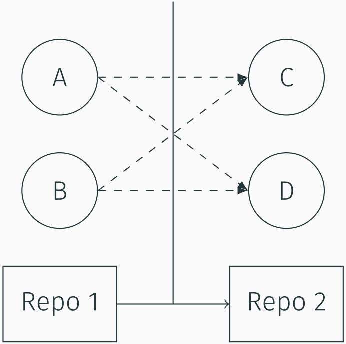


Figure 5: Dependency of repository 1 on repository 2 with the respective developers. Dashed lines indicate implied links between developers.

Again, solid lines are what we **observe**. Dashed lines is what we **infer**.

1. Not reverse causality - dependency use just mildly spatial

Dependent Variables:	contr_n_links	dep_value
Model:	(1)	(2)
Variables		
dist_cat = Samecity(0-1)	2.018***	0.0754***
	(0.0858)	(0.0138)
dist_cat = Agglo(1-50)	0.7344***	0.0805***
	(0.0873)	(0.0127)
dist_cat = Region(50-200)	0.2039***	0.0254***
	(0.0307)	(0.0095)
dist_cat = Shorttrip(200-700)	0.0416***	0.0045
	(0.0101)	(0.0036)
different country same language	-0.1581***	-0.0222***
	(0.0184)	(0.0082)
different country diff language	-0.2476***	-0.0499***
	(0.0322)	(0.0115)
Pseudo R^2	0.86084	0.98866
Observations	3,478,716	3,202,202

Origin, destination city FE, Clustered (city_destination & city_origin) standard-errors in parentheses

1. Not reverse causality - city size

- Adding city size does not matter much

2. + 3. + 3. Selection

- Selection I: Random / assortative matching + large cities having best developers
- No. This would lead to opposite result

- Selection II: best developers select into good projects and search more
- Let us condition on developer quality

- Selection III. High FC for cross-city projects – developers work more
- Let us condition on commits

MORE: ▶▶ More on a sketch of a theory

Results 2: Selection? Partialing out developer quality and commits

Dep.var: N Dependents	(1)	(2)	(3)	(4)
Count of cities	0.2306***	0.2456***	0.1810***	0.2773***
	(0.0491)	(0.0499)	(0.0511)	(0.0532)
Count of countries	0.3057***	0.2856***	0.3251***	0.2856***
	(0.0637)	(0.0649)	(0.0657)	(0.0662)
Constant	1.148***	0.6678***	0.0675	-1.703***
	(0.0857)	(0.1130)	(0.1198)	(0.1572)
Age, N_Dev	Yes	Yes	Yes	Yes
Coder city	No	Yes	Yes	Yes
Coder quality	No	No	Yes	Yes
Commits	No	No	No	Yes
Fit statistics				
Pseudo R^2	0.11532	0.12270	0.14772	0.18401
Observations	36,491	35,679	32,056	32,056

Success and dispersion

- Compare developers of similar quality based in similar locations
- Exclude success driven by bigger spatial reach of developers
- Account for more work per project in dispersed teams

- Group of diverse developers will create more successful projects

Ongoing data work

- Organizations
- Missing city info
- Unlocking developer ethnicity based on names
- Other OSS languages: Python, Ruby, C++, Java, Rust
- ...

Discussion

Summary

- Location matters even for coding

Summary

- Location matters even for coding
- Will the best developers congregate in big cities to create best code?

Summary

- Location matters even for coding
- Will the best developers congregate in big cities to create best code?
- No. Spatially dispersed developers create code that is more widely adopted.

- Sorting matters: good developers write good code used by more. But not explains

Summary

- Location matters even for coding
- Will the best developers congregate in big cities to create best code?
- No. Spatially dispersed developers create code that is more widely adopted.

- Sorting matters: good developers write good code used by more. But not explains

- There is something else...

References

AlShebli, Bedoor K., Talal Rahwan, and Wei Lee Woon, "The preeminence of ethnic diversity in scientific collaboration," *Nature communications*, 2018, *9*, 1–10.

Anderson, James E, Ingo Borchert, Aaditya Mattoo, and Yoto V Yotov, "Dark costs, missing data: Shedding some light on services trade," *European Economic Review*, 2018, *105*, 193–214.

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.

Bahar, Dany, "The hardships of long distance relationships: time zone proximity and the location of MNC's knowledge-intensive activities," *Journal of International Economics*, 2020, *125*, 103311.

Bailey, Mike, Abhinav Gupta, Sebastian Hillenbrand, Theresa Kuchler, Robert Richmond, and Johannes Stroebel, "International Trade and Social Connectedness," *Journal of International Economics*, Mar 2021, *129*, 103418.

Bernard, Andrew B, Andreas Moxnes, and Yukiko U Saito, "Production networks, geography, and firm performance," *Journal of Political Economy*, 2019, *127* (2), 639–688.

Bircan, Cagatay, Beata Javorcik, and Stefan Pauly, "Creation and Diffusion of Knowledge in the Multinational Firm," 2021. Working Paper.

Blum, Bernardo S and Avi Goldfarb, "Does the internet defy the law of gravity?," *Journal of international economics*, 2006, *70* (2), 384–405.

Chaney, Thomas, "The network structure of international trade," *The American Economic Review*, 2014, *104* (11), 3600–3634.

Davis, Donald R and Jonathan I Dingel, "A spatial knowledge economy," *American Economic Review*, 2019, *109* (1), 153–170.

Fackler, Thomas, Michael Hofmann, and Nadzeya Laurentsyeva, "Defying Gravity: What Drives Productivity in Remote Teams?," Technical Report, LMU CRCT Discussuion Paper 427 2023.

Head, Keith, 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, (24-038).

Jaffe, Adam B, Manuel Trajtenberg, and Rebecca Henderson, "Geographic localization of knowledge spillovers as evidenced by patent citations," *Quarterly journal of Economics*, 1993, *108* (3), 577–598.

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

Lerner, Josh and Jean Tirole, "Some simple economics of open source," *The journal of industrial economics*, 2002, *50* (2), 197–234.

Li, Yao Amber, "Borders and distance in knowledge spillovers: Dying over time or dying with age?—Evidence from patent citations," *European Economic Review.*, October 2014, *71*, 152–172.

Stein, Ernesto and Christian Daude, "Longitude matters: Time zones and the location of foreign direct investment," *Journal of International Economics*, 2007, *71* (1), 96–112.

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

Behind Poisson 1: Individual matching decision

Collaboration or dependency link between developer i and j,

$$\Pr(Y_{ij} = 1 | x_i, x_j, d_{ij}) = \Pi(\beta_1 x_i + \beta_2 x_j + \beta_3 d_{ij})$$

with

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

the logistic function

Assumption: Independence across links, add fixed effects

Behind Poisson 2: Aggregate to Poisson

In practice, distance only varies at the city level. Take origin city o and destination city d.

$$Y_{od} := \sum_{i \in o} \sum_{j \in d} Y_{ij}$$

$$\Pr(Y_{od} | x_o, x_d, d_{od}) = \text{Binomial}[N_o \times N_d, \Pi(\beta_1 x_i + \beta_2 x_j + \beta_3 d_{ij})]$$

Here $N_o \times N_d$ is the total number of *potential* links between cities o and d.

When Π is small, we aggregate i into cities o, and j into cities d

$$\Pr(Y_{od} | x_o, x_d, d_{od}) \approx \text{Poisson}[N_o \times N_d \times \exp(\beta_1 x_o + \beta_2 x_d + \beta_3 d_{od})]$$

Behind Poisson 3: Having exposure is key

We may also look at a subsample (like users not in the same GitHub organization)

$$Y_{od,\text{not org}} := \sum_{i \in o} \sum_{j \in d, j \notin \text{org}(i)} Y_{ij}$$

This changes the *exposure variable*,

$$\Pr(Y_{od,\text{not org}} | x_o, x_d, d_{od}) \approx \text{Poisson}[N_{od,\text{not org}} \times \exp(\beta_1 x_o + \beta_2 x_d + \beta_3 d_{od})],$$

with $N_{od,\text{not org}}$ the number of user pairs in city o, d, *not sharing* an organization.

Important: $N_{od,\text{not org}}$ may be zero.

What is a Poisson regression?

First-order conditions for Maximum Likelihood:

$$\sum_o \sum_d x_i [Y_{od} - N_{od} \exp(\beta_1 x_o + \beta_2 x_d + \beta_3 d_{od})] = 0$$

$$\sum_o \sum_d x_j [Y_{od} - N_{od} \exp(\beta_1 x_o + \beta_2 x_d + \beta_3 d_{od})] = 0$$

$$\sum_o \sum_d d_{ij} [Y_{od} - N_{od} \exp(\beta_1 x_o + \beta_2 x_d + \beta_3 d_{od})] = 0$$

- Level (not log) error terms are orthogonal to RHS variables.
- Exposure variable has fixed exponent of 1 ($\approx$ weighting).
- Standard errors computed from GMM, not ML. E.g., we allow for two-way city clustering.

What is an observation?

Two interpretations:

1. 10 billion potential developer pairs
2. 3.7 million city pairs

Model sketch

- Production of code is driven by utility gains of creating code used by many people
- Developers are heterogeneous in coding quality.
- Developers collaborate with others when
  - Task is too complex for a single person. Economies of scale.
  - ...
- There is selection into projects: best developers write most complex packages.

Model sketch 2: The role of geography

- Developers are dispersed geographically – located in a discrete set of N_c cities
  - City size (number of developers) Pareto distributed
  - Size may be driven by first geography (later), such as proximity to University, tech firms or the beach.

- Heterogeneity of developers: at every location, their distribution is Pareto

- Random matching: simple random selection of collaborators
- Assortative matching: Developers match with developers of same quality

Model: self selection of developers

- If best programmers are in big cities (Pareto with different k across cities): size and quality correlated
- Top developers coming from large cities will produce best code –> more popular code.
- Best code will come more than proportionally from large cities

- Assortative matching reinforces this aspect, as big city developers will only work with big city developers

- Best code written by people in top cities (like SF) – homogeneity

Model: There are search costs

- Costs of setting up a partnership and maintaining it
- Search costs of inputs (code chunks)
  - Written together – finding a collaborator
  - Using already published code – finding a package
- Search costs vary with distance – lower inside the city

Model: developer heterogeneity

- There is a set of possible coding skills, S
- Developers randomly vary in each skill, $s = 1, 2, 3...S$
- Two developers who are on average same quality still have difference and can benefitb from collaboration, where the pair's skill is max

Model: Dispersion forces

- Developers differ to some extent, and so search is needed
- There is a search cost, higher for other cities
- Better developers pay higher search cost and hence can search a larger pool across cities

Model: Additional aspects

- Face to face matters when creating complex projects.
- Some cities specialize in some tasks

Bugs 🐞

Model

Long-standing question in economics: how does competition affect innovation?

Model the special features of the OSS market.

Special features

1. Price is zero. Only compete in quality.
2. Software projects often start as a developer's own need.
3. Quality is only partly observable.
4. Collaboration is important.

Outline

1. Defining software quality
2. Producing quality
3. The market for software
4. Testable predictions
5. First evidence from GitHub

Quality

Software quality

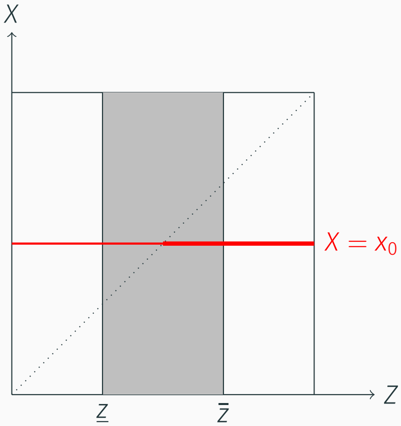
Users have a use case X.

Developers write code $\bar{z}$ and tests $\underline{z}$. Software quality is random $Z \sim U[\underline{z}, \bar{z}]$.

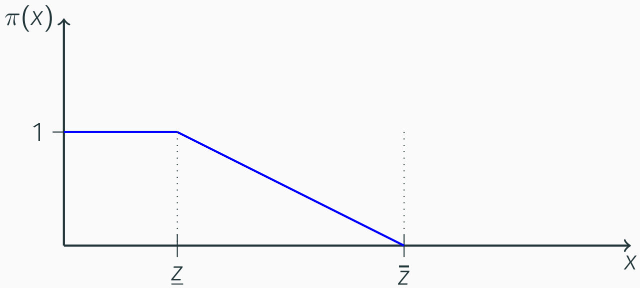
Software only works if $Z > X$.

$$\Pr(Z \text{works for} X) := \pi = \frac{\bar{z} - X}{\bar{z} - \underline{z}}.$$

Software quality



Probability of software working for a given use case



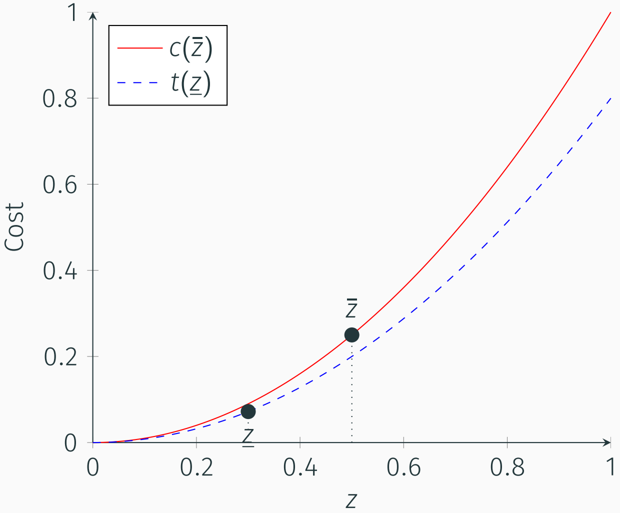
The production of quality

Coding up to $\bar{z}$ costs $c(\bar{z})$. Increasing and convex.

Testing up to $\underline{z}$ costs $t(\underline{z})$. Increasing and convex.

(Current results for $t(z) = \tau c(z)$ with $\tau \leq 1$.)

Cost of quality



Market

Three market environments

1. Do-it-yourself: developer writes code for own use. $X = u$ is known.
2. Shared platform: developer writes code for others. $X \sim F$ is unknown.
3. Competition: n developers write code for the same set of users.

The DIY economy

The developer maximizes

$$\max_{\underline{z},\bar{z}} \frac{\bar{z}-u}{\bar{z}-\underline{z}} - t(\underline{z}) - c(\bar{z})$$

subject to $\underline{z}, \bar{z} \geq 0$ and $\underline{z} \leq \bar{z}$.

The platform economy

Assume developer can capture $\phi \ll 1$ share of the value of the software.

She maximizes

$$\max_{\underline{z},\bar{z}} \phi \int \frac{\bar{z}-x}{\bar{z}-\underline{z}} dF(x) - t(\underline{z}) - c(\bar{z})$$

subject to $\underline{z}, \bar{z} \geq 0$ and $\underline{z} \leq \bar{z}$.

Competition

Two-sided market with U users and D developers.

Each user meets n developers at random.

They choose the software with the highest $\underline{z}$.

Competition

With $G(z)$ is the distribution of tested software quality in the marketplace,

$$\Pr(z_j \text{ wins} | x_i, \underline{z}_j, n) = G^{n-1}(\underline{z}_j),$$

Developer's problem

Maximize

$$\max_{\underline{z},\bar{z}} \frac{\phi n U}{D} \int \frac{\bar{z} - x}{\bar{z} - \underline{z}} dF(x) G^{n-1}(\underline{z}) - t(\underline{z}) - c(\bar{z})$$

Collaboration

Collaboration helps overcome diminishing returns to coding. With n collaborators, the total coding cost up to $\bar{z}$ is

$$C(\bar{z}) := \min_{\{z_i\}} \sum_{i=1}^{n} c_i(z_i) \text{ s.t. } \sum_{i=1}^{n} z_i \geq \bar{z}$$

$$nc(\bar{z}/n) < c(\bar{z})$$

There may be increasing returns to collaboration: lower marginal cost $\rightarrow$ higher demand $\rightarrow$ more individual contribution.

Predictions

Predictions on testing

1. DIY projects are not fully tested.
2. Shared projects are.

Predictions on code quality

1. Standalone projects are limited by developer's own need. Diminishing returns to quality.
2. Shared projects have higher quality. Constant returns to quality.
3. Competition increases quality. Increasing returns to quality.

Predictions on collaboration

1. Collaborative project may have *more* individual contribution.
2. Especially in shared projects.

Measurement

Six biggest languages on GitHub: JavaScript, Python, Java, Ruby, PHP, and C++.

Contribution: number of commits per developer per project.

Compare the *same* developer in the *same* language across projects.

Developer skill: average number of stars per solo-authored project.

Good developers contribute more to shared projects

VARIABLES	(1) Private projects	(2) DIY projects	(3) Shared projects	(4) Popular projects
Developer skill	0.0101***	0.00840***	0.0867***	0.110***
	(0.00108)	(0.00126)	(0.00195)	(0.00362)
No. contributors (log)		0.0450***	0.0265***	-0.0680***
		(0.00388)	(0.00478)	(0.00638)
Constant	3.233***	3.197***	3.125***	3.243***
	(0.00281)	(0.00326)	(0.00442)	(0.0134)
Observations	361,196	629,039	514,259	136,503
R-squared	0.002	0.002	0.038	0.037

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

Popular projects attract better developers

VARIABLES	(1) Commits	(2) Commits	(3) Commits
Shared on a platform (dummy)	0.0731***	0.0457***	0.0281***
	(0.00789)	(0.0108)	(0.0107)
Has downstream projects		0.0370***	0.0314***
		(0.0100)	(0.00998)
Has 5 or more stars (dummy)			0.116***
			(0.00775)
Constant	3.055***	3.054***	2.889***
	(0.0112)	(0.0113)	(0.0163)
Observations	172,495	172,495	172,495
R-squared	0.680	0.680	0.681

Robust standard errors in parentheses

*** $p<0.01$, ** $p<0.05$, * $p<0.1$