Bugs 🐞

Miklós Koren (CEU, HUN-REN KRTK, CEPR and CESifo), Gábor Békés (CEU, HUN-REN KRTK, and CEPR), Julian Hinz (Bielefeld University and Kiel Institute for the World Economy)

March 6, 2024[1]

[1]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.

Software is eating the world

The weightless economy

"Software is eating the world." (Andreessen, 2011)

Open-source software (OSS) is everywhere

Linux, Apache, MySQL, PHP, Python, R, Julia, Android, Firefox, Chrome, etc.

Also included in proprietary software

Two economic puzzles in open source

Why do people work for free?

Altruism, reputation concerns, alternative business models. Sizeable economic literature.

Two economic puzzles in open source

Why do people work for free?

Altruism, reputation concerns, alternative business models. Sizeable economic literature.

How can a bunch of amateurs produce quality software?

Salient features of OSS

Price is zero

Not even that unique.

Salient features of OSS

Price is zero

Not even that unique.

Scratch your own itch

Developers are often their own first users: grep, TeX, Linux, git, etc.

Free access to source code

"Given enough eyeballs, all bugs are shallow." (Raymond, 1999)

Software quality is only partly observable

Testing is important.

Based on two studies

Success and geography in the weightless economy: Evidence from open-source software

Békés, Hinz, Koren, and Lohman. 2024.

Bugs 🐞

Koren, Békés, and Hinz. 2024.

Data

GitHub

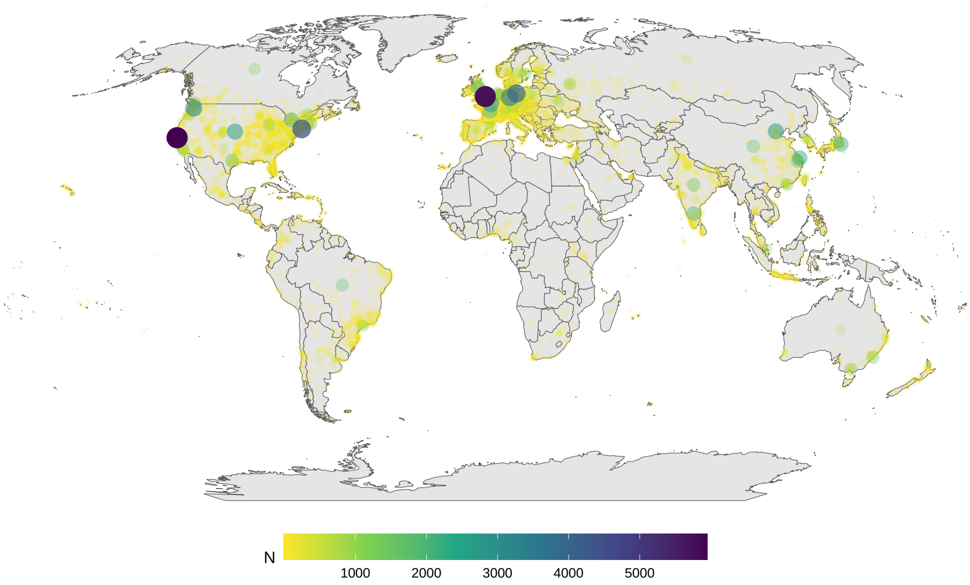
Snapshot of all public repositories on GitHub on 2019-06-01. Six largest languages: JavaScript, Python, Java, Ruby, PHP, and C++. Drop smallest and largest projects. 4.4m projects, 2.7m users. Self-reported location.

libraries.io

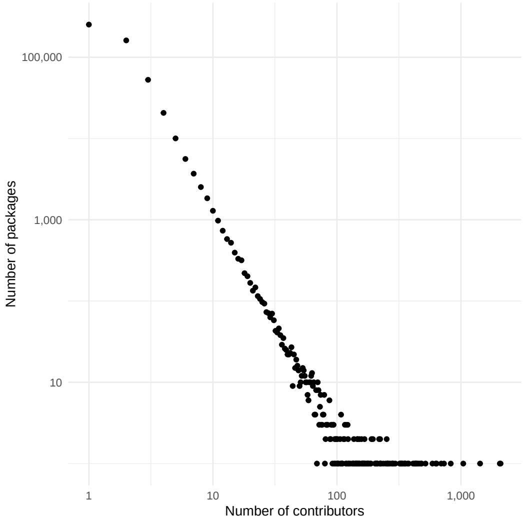
Dependency data for projects on major package managers (npm, PyPI, Maven, RubyGems, etc). Studying npm (JavaScript) today.

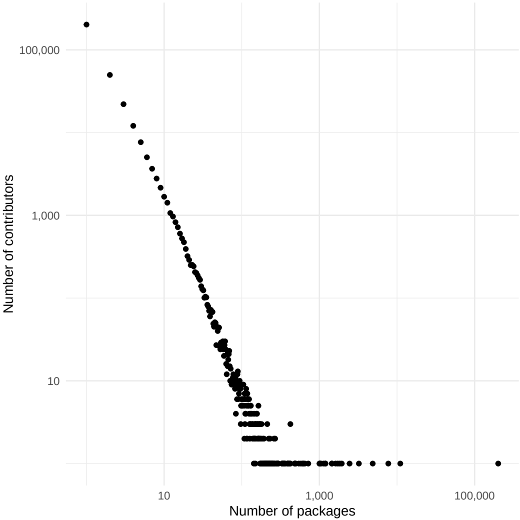
Success and geography in the weightless economy: Evidence from open-source software

Developer density around the globe

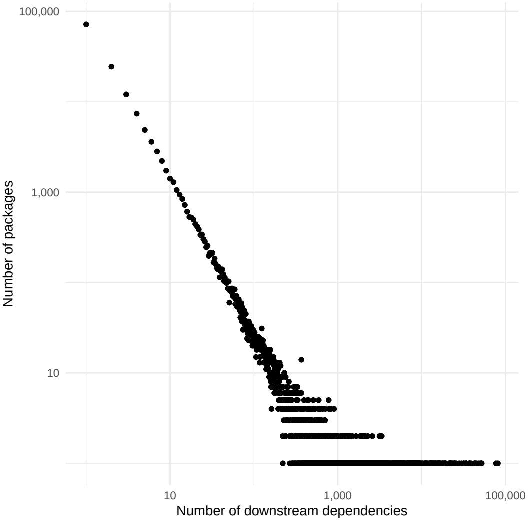


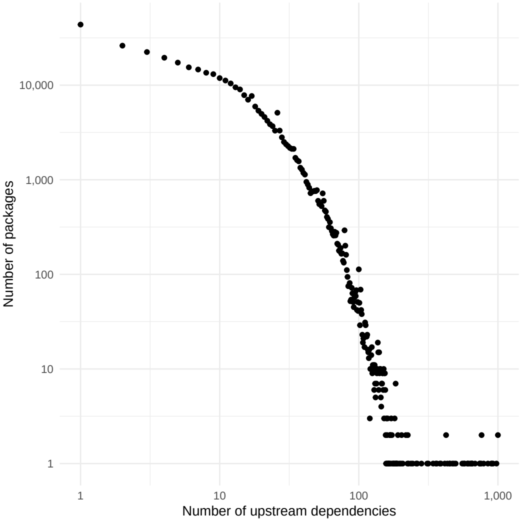
Large variation in number of projects and developers



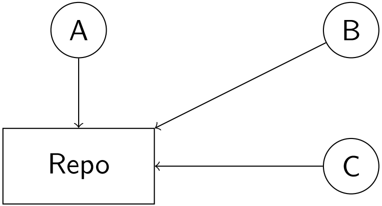


With limits on how many projects one imports

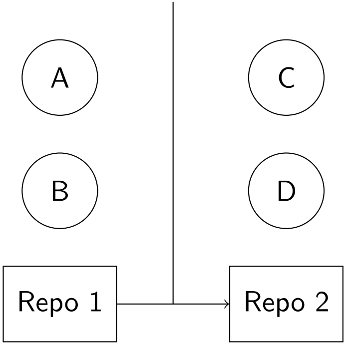




Measuring collalboration and dependencies



(a) Developers committing to a repository.



(b) Dependency of repository 1 on repository 2 with the respective developers.

Gravity model of collaboration

Developer i and j collaborate with probability

$$\Pr(\text{Collaboration}_{ij}) = \exp(\alpha x_i + \beta x_j - \gamma \times \text{distance}_{ij})$$

Aggregate across city pairs d and o:

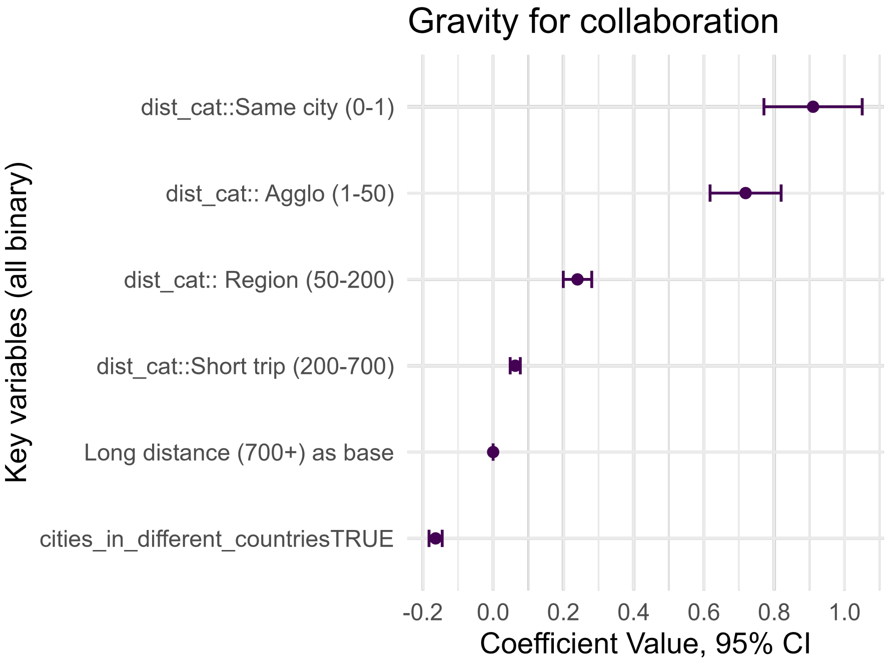
$$E(N_{do}) = \exp(\alpha x_d + \beta x_o - \gamma \times \text{distance}_{do})$$

Estimate this with Poisson maximum likelihood.

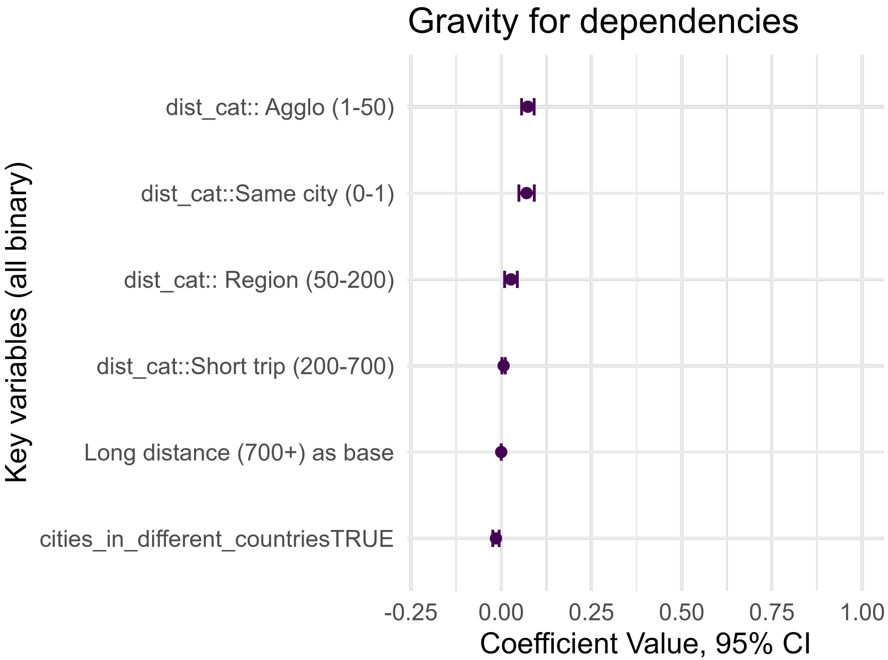
Three margins of collaboration

1. Committing to the same project
2. Importing someone else's project
3. Members of the same organization

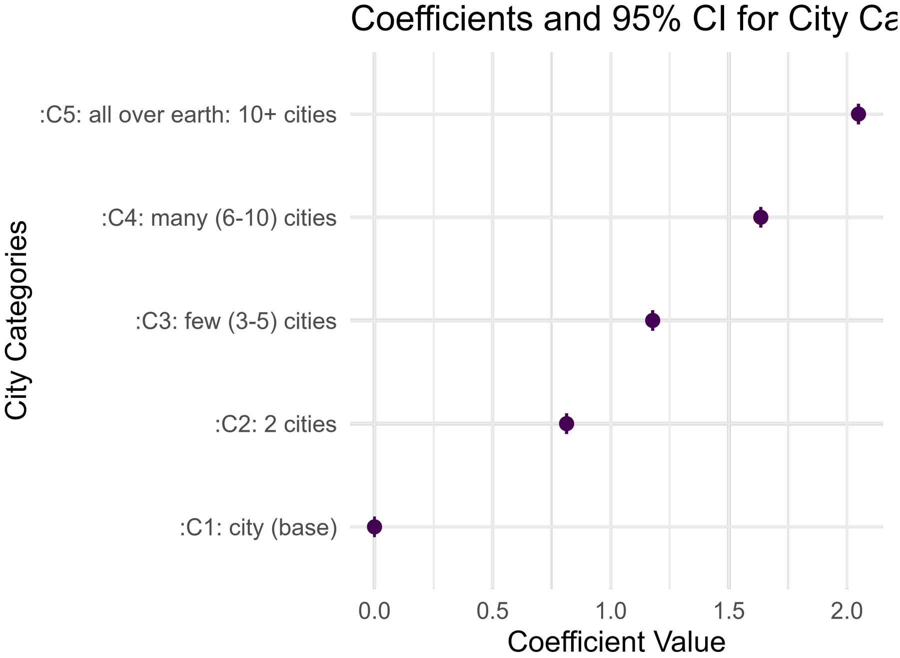
Strong localization of collaboration patterns



No localization of dependencies



Diverse teams produce more popular software



Additional results

Organizations help overcome distance. Almost no distance penalty for developers within the same GitHub organization.

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

Quality

Software quality

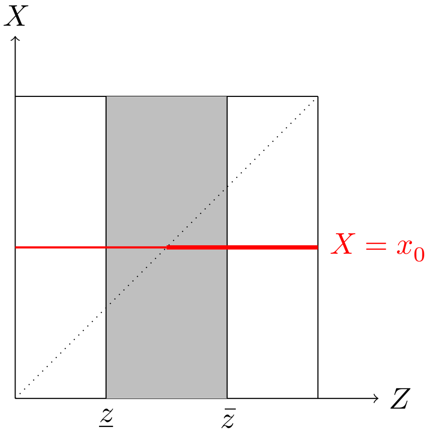
Users have a use case X.

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

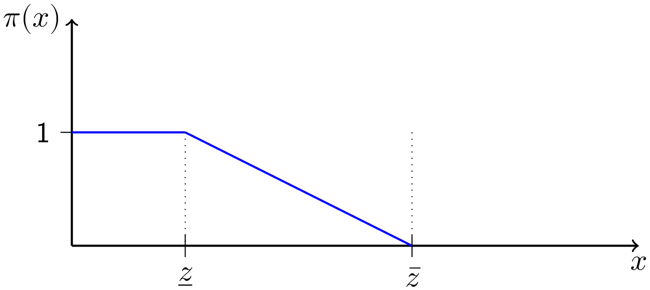
Software only works if $Z > X$.

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

Software quality



Probability of software working for a given use case



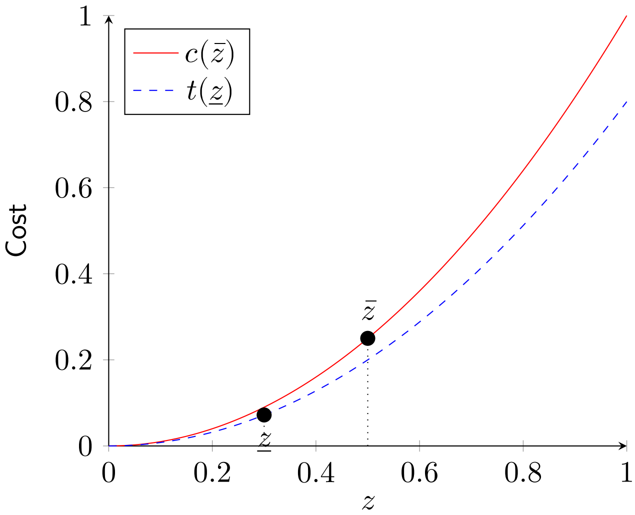
The production of quality

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.

Larger projects are written by more people

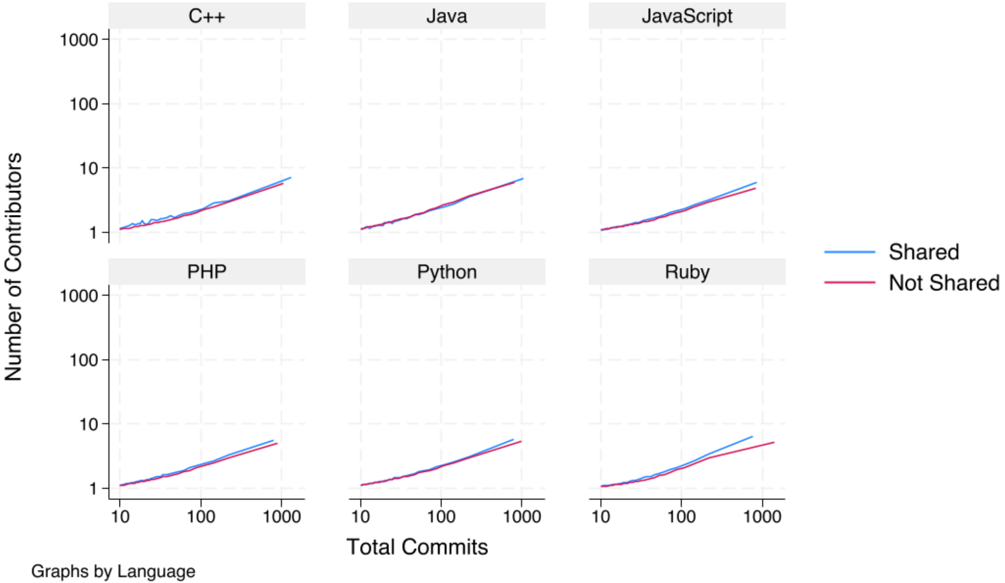


Figure 2: Larger projects by more people

Larger projects are more popular

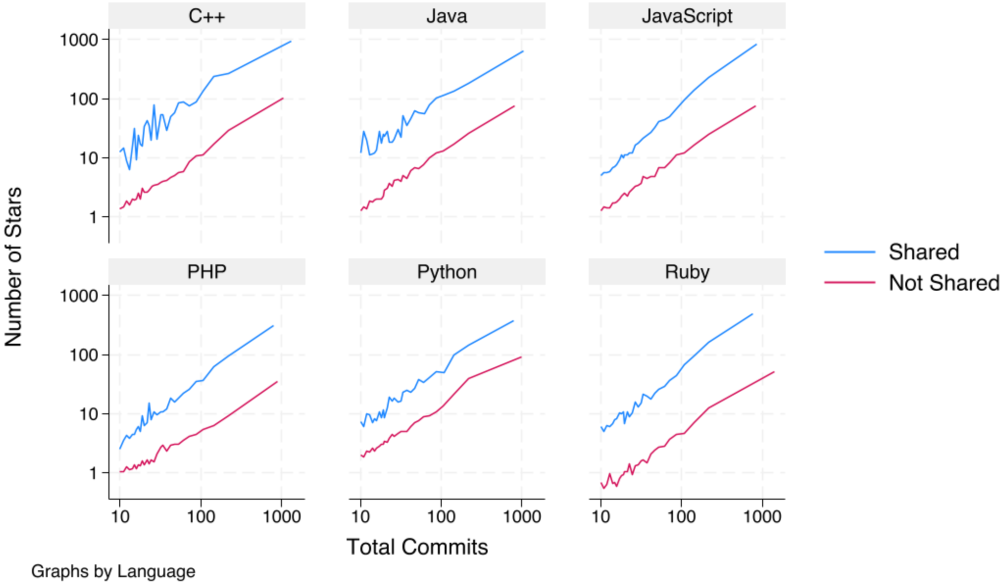


Figure 3: Larger projects are more popular

Larger projects have more bug discovery

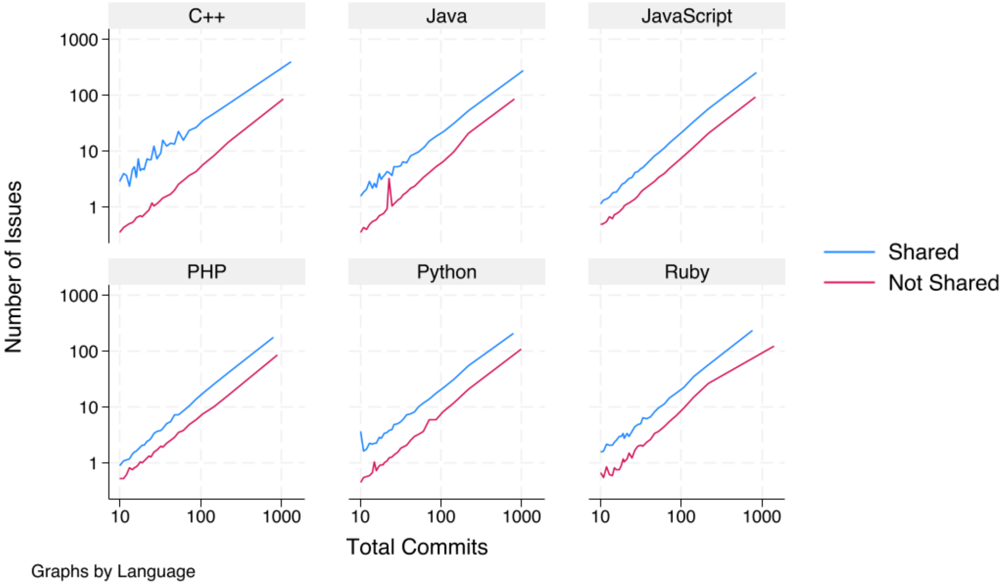


Figure 4: Larger projects have more bug discovery

Larger projects solve a larger share of issues

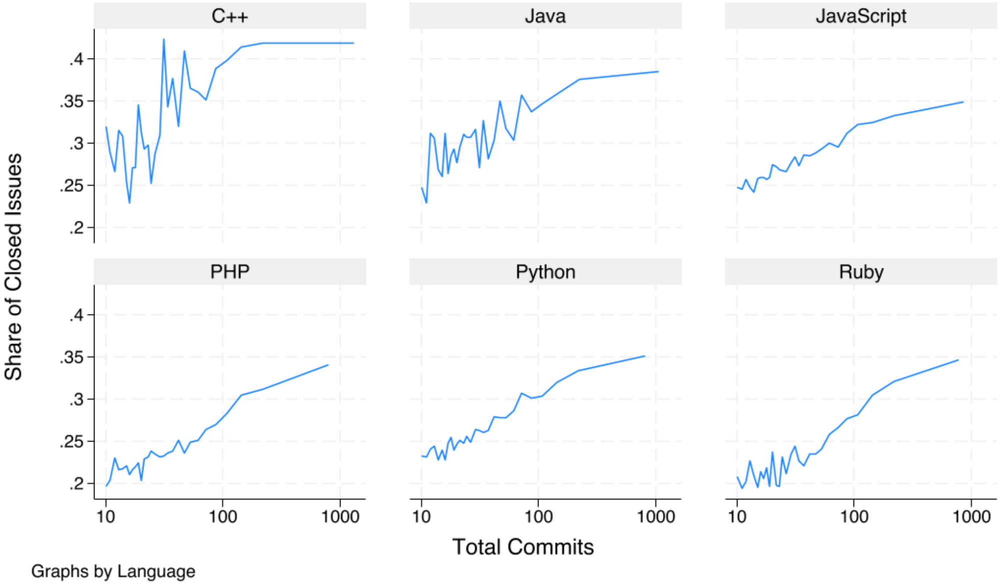


Figure 5: Larger projects solve a larger share of issues

Better developers contribute more to shared projects

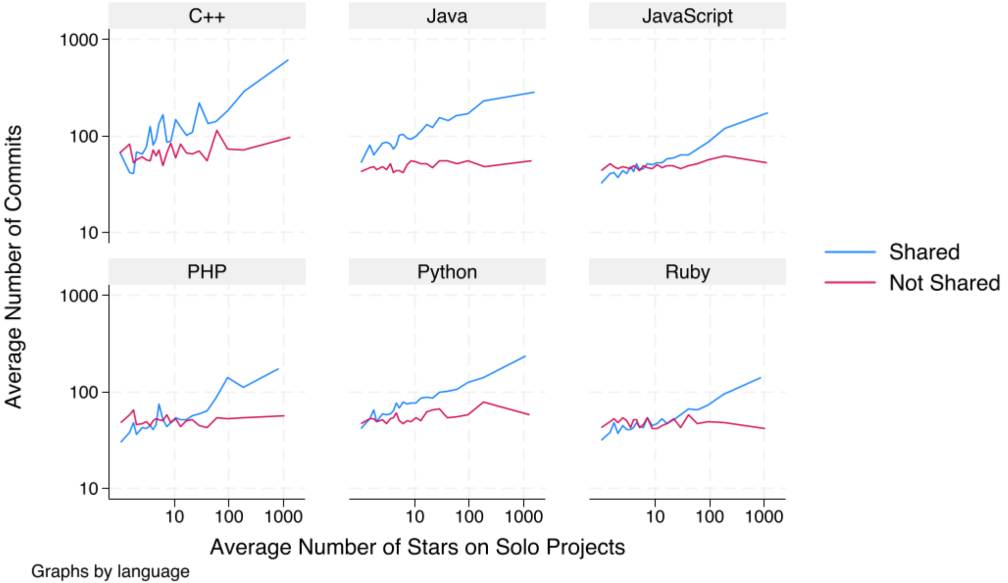


Figure 6: Better developers contribute more to shared projects

Shared projects are better quality

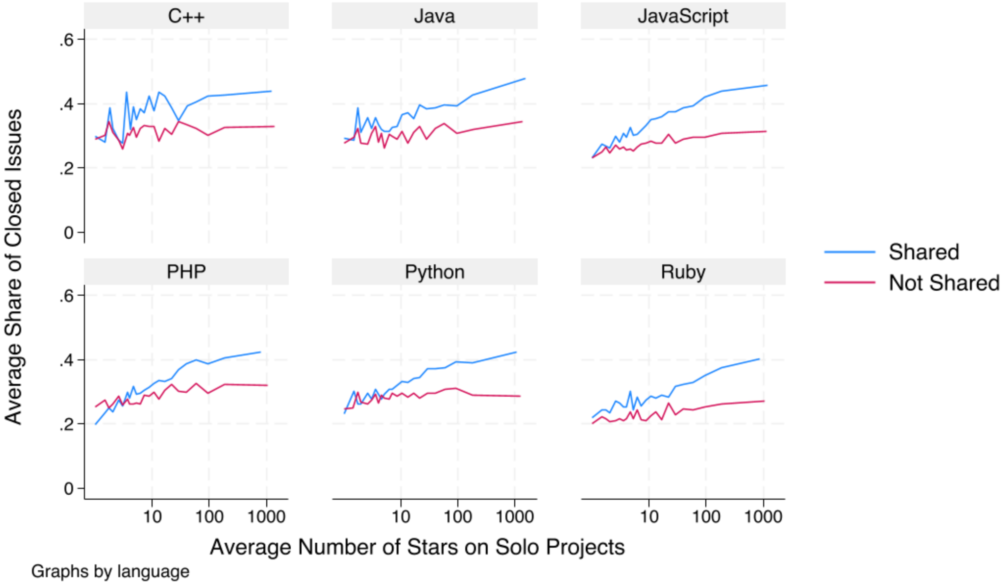


Figure 7: Shared projects are better quality

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

Next steps

Measure test coverage.

Interaction with users: bug reports, feature requests.

Sorting into collaboration.

Get in touch

@korenmiklos

@gaborbekes

@julianhinz