0.1 Background

In my empirical analysis, I examine how changes in the cost of knowledge acquisition and communication across the hierarchy affect hierarchical structure and the work of OSS contributors. I study the impact of increases in the cost of knowledge acquisition by examining knowledge turnover caused by contributor departures. I study the impact of decreases in communication cost by examining the impact of issue and pull request template adoption by projects.

1. Contributor Turnover
2. Issue & Pull Request Templates

0.2 Knowledge Turnover

Question: What is the causal impact of knowledge turnover on OSS hierarchical structure and contributor time allocation?

0.2.1 Context

One of the major worries with OSS development is the reliance of projects on a few contributors, who, without a contractual obligation of a project, may disappear at a moments notice and bring with them the knowledge and skills necessary for key components of, or in some cases, the whole project, to function. One way to measure variation in the cost of knowledge acquisition in projects is to exploit OSS developer turnover. When a key contributor for an OSS project leaves, it is much harder for new contributors to acquire knowledge about projects because a source of information is gone.

It would be good for me to explore some examples of this in the data, so I can tell an interesting story about the impact of contributor turnover. Some initial questions

- How were areas where this contributor was involved in affected deferentially compared to other areas?
- Do we observe evidence of reduced discussion in issues related to a PR (less learning?)
- Do we observe evidence of less PR review comments & slower PR reviews?
- Do we observe evidence that future PRs by contributors who would be affected are rejected at higher rates?

0.2.2 Adoption and Treatment

1. We can consider variation in turnover by examining the "truck" factor of projects, which measures the number of key members. There are also variants to this measure, such as looking at the identity of the person who wrote the last line of code, as opposed to total lines written per person.

2. I should also control for the presence of knowledge preservation tools. The impact of contributors leaving will be lessened by the presence of good documentation. For example, if a departing contributor wrote a lot of documentation count, that means that they probably did preserve their knowledge for future contributors so their departure will not impact knowledge acquisition by as much. I can measure this by looking at the amount of code for '.md', '.rst' or other documentation files types that they wrote.

0.2.3 Good Descriptive Facts

- What is the rate of turnover? Contextualize this with information about average project size.

- What does the distribution of importance (for departing contributors) look like? Can contexualize using a variety of measures such as LOC written (truck factor), issues commented on, PRs reviewed, etc

0.2.4 Predicted Empirical Effects - Hierarchical Structure

I am interested in three metrics related to organizational structure

1. Span: Ratio of higher to lower ranked contributors
   Garicano's model predicts that there will be more write rank contributors
   What does my data show?

2. Frequency: % of all problems solved at each layer
   Garicano's model predicts that more problems will be solved by highly ranked contributors
   What does my data show?

3. Output: How many problems are being solved
   Garicano's model predicts that overall production will decrease.
   What does my data show?

On span: Since some organizations don't actively manage hierarchies, I may want to filter by "activeness" (separately, there may be something interesting to say about "activeness" versus actual hierarchy, along the lines of security). I may also want to just consider the numerator (write rank contributors) as the project can only control the quantity of promotions, not new contributors.

On frequency: I define a problem as a pull request, although not all pull requests are created equal. In this case, I can weight pull requests by lines of code, files changed (removing moved files) or other measures (that I think of). Separately, since the flow of read-rank contributors cannot be controlled, I may want to just consider the % of all problems solved by write rank contributors. I can start by assuming problems are solved individually and validate this by examining the proportion of code written/PR by one individual. It will also be interesting to break down where the changes in % problems solved are coming from.

On output: This is my opportunity to connect organizational structure outcomes to OSS development outcomes such as quantity or % of PRs merged, and % of opened issues that are closed. There may also be non-code related development metrics of interest. What are outcomes that organizations care about?

0.2.5 Predicted Empirical Effects - Contributor Characteristics

I am interested in three metrics related to individual conrtibutors

1. Individual Output: How many problems is each contributor solving?
   Garicano's model predicts that each read rank contributor will solve less problems and write rank contributor will solve more
   What does my data show?

2. Individual Skill: How knowledgeable are they?
   Garicano's model predicts that each read rank contributor will lose skill and write rank contributors will gain skill
   What does my data show?

3. Individual Value Added: What is the value of the problems they're solving Garicano's model doesn't have a prediction for this
   I don't have a prediction for this either but I think it would be fun to answer. It's probably not the most important thing to spend time on though.

On individual output: I may encounter measurement issues due to integer constraints with "problems". This is a good place to incorporate ideas relating to problem

weighting from the hierarchical structure section.

On individual skill: Some examples of skill measures are the length of time required to solve a problem, the number of LOC someone has written (for a project), their quantity of GH badges. The problem is that problem difficulty is an unobserved confounder. Two fun ideas to measure problem difficulty are how close ChatGPT can answer the question and the cyclomatic complexity of the problem (former is hard to implement, latter has endogeneity issues).

On value: One fun idea is to examine the text that's commonly observed in SO for this python library, and see how similar it is to a PR/issue - high value indicates high similarity (solving something people are asking about).

0.2.6 Literature

Economics

OSS

Rashid, Clarke, and O'Connor 2017 has a fairly good literature review on knowledge loss in OSS. Nassif and Robillard 2017 provides interesting comparisons about different statistics for measuring knowledge loss.

Some papers that I'm hoping to read are

1. https://ieeexplore.ieee.org/abstract/document/8870181?casa_token=1rqBePn8XnoAAAAA:wCuI CNViaN8n-_20B3PaSKabeFN2vp5ZX
2. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4755634 (need to cite)
3. https://doras.dcu.ie/29119/1/RashidIJIMPaper-RevisionFollowingPeerReview.pdf (need to cite)
4. Managing knowledge assets for open innovation: a systematic literature review (need to cite)
5. Exploring the Foundations of Cumulative Innovation: Implications for Organization Science (need to cite)

0.3 Issue & Pull Request Templates

0.3.1 Summary

In February of 2016 (https://github.blog/?s=issue+templates), GitHub released issue and pull request templates. GitHub stated that "now project maintainers can add

templates for Issues and Pull Requests to projects, helping contributors share the right details at the start of a thread". This is important because it ensures that OSS contributors have completed required checklist items before opening an issue/PR, and also makes it easier for reviewers to identify to find particular key pieces of information.

0.3.2 Adoption and Treatment Notes

I can measure initial adoption of PR and Issue templates by looking at when an ISSUE_TEMPLATE (optional ".md" extension?) file or PULL_REQUEST_TEMPLATE (optional ".md" extension?) was created. I can compare projects that adopted these templates during similar timeframes because while the particular timing of adoption might be adoption, adoption within a certain timerange tells us a lot about the attitudes of the OSS contributors towards improving their projects. In May 2018, users acquired the option to choose a template from (presumably) a variety of options when making a new issue (source). However, while this implies that there may be multiple treatments, this is still easier to analyze because there are less possible combinations (less templates) and adoption is more spread out throughout time. Are there any interesting stories I can tell wrt to issue/pr templates?

0.3.3 Predicted Empirical Effects - Hierarchical Structure

I am interested in three metrics related to organizational structure

1. Span: Ratio of higher to lower ranked contributors
   Garicano's model predicts that there will be more write rank contributors

2. Frequency: % of all problems solved at each layer
   Garicano's model predicts that more problems will be solved by highly ranked contributors
   What does my data show?

3. Output: How many problems are being solved
   Garicano's model predicts that overall production will increase
   What does my data show?

0.3.4 Predicted Empirical Effects - Contributor Characteristics

I am interested in three metrics related to individual conrtibutors

1. Individual Output: How many problems is each contributor solving?
   Garicano's model predicts that each read rank contributor will solve less problems and write rank contributor will solve more
   What does my data show?

2. Individual Skill: How knowledgeable are they?
   Garicano's model predicts that each read rank contributor will lose skill and write rank contributors will gain skill
   What does my data show?

3. Individual Value Added: What is the value of the problems they're solving
   Garicano's model doesn't have a prediction for this
   I don't have a prediction for this either but I think it would be fun to answer. It's probably not the most important thing to spend time on though.

0.3.5 Literature

Some papers that I'm hoping to read are

Economics

1. Organizational Responses to Product Cycles
   The main contribution of this paper is to show how organizations respond when problems become more complex but their capacity to expand is constrained. The setting they consider is automobile manufacturing, which is affected by product cycles. Every few years, a new car model is released and producing it introduces complex problems, because the factory is new to the production process for that model. The theory of hierarchy predicts that when problems get harder, more hierarchical layers are introduced because workers on lower layers are able to solve a smaller proportion of of problems. The authors observe in the data that the number of layers actually reduces, as the firm trains middle-layer workers to become higher-layer workers, and then postpones the filling of open middle-layer position. This brings lower-layer workers involved in production closer to higher-layer workers, who have been trained to now have the knowledge to handle these increasingly complex problems. When problem complexity is reduced, middle-layer positions are filled. They develop a new theory (which I have to read) to support this.

2. Face-to-Face Communication in Organizations

3. The Anatomy of French Production Hierarchies
4. Firm Organization with Multiple Establishments
5. Training, Communications Patterns, and Spillovers Inside Organizations
6. The World Management Survey at 18: lessons and the way forward
7. Management as a Technology?
8. The Incentive Effect of Scores: Randomized Evidence from Credit Committees
9. Returns to On-the-job Soft Skills Training

OSS

1. An Empirical Analysis of Issue Templates Usage in Large-Scale Projects on GitHub
2. Consistent or not? An investigation of using Pull Request Template in GitHub
3. To Follow or Not to Follow: Understanding Issue/Pull-Request Templates on GitHub
4. An empirical examination of newcomer contribution costs in established OSS communities: a knowledge-based perspective

1 Specific Roadmap

- Open source software development on GitHub takes place within a hierarchy
- Why do we need a hierarchy? On GitHub, it's because not everyone is suited to handle all tasks. For example, we don't want any programmer to be able to change the codebase by approving a pull request.
- Focusing on pull requests, there are likely many reasons why we don't want any programmer approving pull requests, but suppose that all programmers are well-intentioned. Even so, we don't want any programmer approving pull requests because they might not have the knowledge to catch mistakes or the time to check whether the code does everything it claims to. I define this as their ability to handle certain tasks for brevity.

- On GitHub, we have a hierarchy where contributors are organized onto different ranks. Mechanically, higher ranked contributors only differ from lower ranked contributors in having a large choice set of actions.
- This leads me to my descriptive analysis. My first question is how do OSS contributors differ across ranks? In particular, I'm interested in the following characteristics
  - How much time do they spend on a project (can I use the "average 13 hours on contributing" technique)? This captures **effort**
  - How do they allocate their time in a project? (straightforward - PRs, issues, issue comments, pushes, PR reviewing)
  - What is their ability?
    Various measures of contributor skill derived from measures unrelated to "problem solving": # of GH followers, # of GH achievements, % of codebase authored, # YOE on GitHub, # YOE in the project, # of merged PRs in other projects (weighted by stars), # of merged PRs in this project. These are absolute measures of skill, although I can make all of them relative by adding project fixed effects.
    # of merged PRs in other projects (weighted by stars) and # of merged PRs in this project worry me because they're affected by problem difficulty - what if someone only chooses to solve hard problems. More broadly, I'm not a huge fan of any of these because I want something that I can observe "evolve" as a direct consequence of time investment someone puts in learning about the project, and none of these fit that category - one can define this as "project-specific problem solving ability" which is related to but not exactly the proxies for skill I've defined here. I haven't been able to come up with a proxy for skill that can evolve and is not confounded by unobserved problem difficulty or is not a direct measure of output. Some of the biased ones are % of PRs that are merged, # of commits made after a PR is opened, % of commits made by PR reviewer,
  - How hard are the tasks they're solving (conditional on task identity)?
    Various measures of problem difficulty: time for PR to be closed/merged, # of PR review comments or # of comments, # of reviewers, # of LOC, # of commits, length of text in a PR description, length of corresponding linked issue, # of files changed, # of PR tags, # of PR reviewers who approved. Note that skill might bias some of these metrics downwards.

We can also regress the probability a problem is assigned to a high ranked contributor on problem characteristics and evaluate what characteristics are positively associated with a higher ranked contributor being assigned. If we assume highly ranked contributors are assigned to difficult problems, we can observe what characteristics determine problem difficulty.
Similarly, if we control for problem difficulty, we can see what problem solution characteristics are positively associated with a high ranked contributor (who is skilled).

- Note that effort, skill and problem difficulty metrics are interrelated. If someone's handling very little tasks, is it because those are hard tasks, they lack skill, or they're putting in little effort?

- I can simplify my setting by making some assumptions

    1. For example, one assumption (that admittedly I am not a big fan of and don't have a clear idea of how to make) is to simplify the distinction between skill and problem difficulty. For example, we know problem difficulty is the lower bound estimate for skill. Perhaps we can make it the upper bound too?
    2. I don't need to establish absolute measures of skill across projects (although it would be great if I could do so). I only need to compare relative levels of skill held within a project and over time.

- My second question is what does the OSS organization look like? I'm interested in the following questions (and more)

    – On span: Ratio of higher to lower ranked contributors at each rank

        1. Start broadly by focusing on the whole hierarchy. What are general project-level trends I see?
        2. How do these trends change when I focus on active contributors?
        3. What if I observe just # of contributors at each rank?
        4. What if I split up contributor types by action (Read rank: whose opening PRs vs. whose opening issues vs. whose also commenting on other people's issues?, Higher rank: whose reviewing & merging vs. opening PRs?)

    – On output: How many and what types of problems are being solved by the project

1. How many issues are being opened, and how many are being resolved
2. How many PRs are being opened and how many are being closed, merged or left open
3. What is issue resolution time? What about PR resolution time?
4. How many reviewers/assignees are assigned to each PR/issue? What % of them actually participate? How much do they participate (conditional on participation)?
5. How long is each issue comment? Issue?
6. How much code is in each PR (commits, LOC, additions/deletions).

- Now that we've established the differences between contributors and what the organization looks like, we want to understand how the organization and contributors change as a consequence of two specific shocks: changes in knowledge acquisition costs and changes in communication costs.
- Flesh out empirical strategy, etc
- First, on organizational structure:
  1. How does the hierarchy change (span)?
  2. How does output change?
  3. How do promotion requirements change? Levels of skill is the primary measure, but experience or past output could also be of relevance.
- Second, on contributor behavior
  1. How does individual contribution behavior change? Effort, skill and problem difficulty encountered. Problem difficulty is relevant because maybe people don't try to solve hard problems as much.
  2. How does the contribution behavior of newer project individuals change?
  3. A really cool figure would be a timeline of how a contributor's skill/problem difficulty evolves over time.
  4.
- Now, for the theory - the goal is to see to reconcile my results with the hierarchy literature. Are the hierarchy models predictions borne out in the data and if not, what puzzles do we need to address?

- A few additional puzzles I need my model to also address:
    1. Why do top contributors code? (perhaps examine really big vs. smaller projects - this might be an organizational flexibility constraint)
    2. How do organizations adapt when people don't have the ability to be promoted? Do they not promote or do they lower their standards?
    3. How do the predictions play out with contributors who have much more freedom, and organizations who have much less choice (aka the OSS setting)?

Are they using the issue template whose being tagged

2 Other questions

1. How does task distribution efficiency change? How does task choice change?
2. What differences do I observe b/w corporate and non-corporate OSS?
3. Control for software project maturity, individual length of affiliation with project
4. What % of assignees end up writing code/participating in discussion?
5.