H2: The nuanced relationship between developer repeated collaboration and project success: A moderate level of internal cohesion within a project is better for a project’s success than very high or very low levels of internal cohesion (Singh et al. 2011).

DVs:

Project success: The number of CVS commits for project i in j years after network construction date t.

For instance, for the network constructed in November 2004 and j =1, we count the number of CVS commits for a project from November 2004 to November 2005 (Singh et al. 2011).

Indep Vs:

Developer Repeat ties as a measure of internal cohesion (adapted): We counted the total number of common repositories up until the date on which each pair of repository developers works or has worked.

Repository Repeat ties: We divided the number of sum of Developer Repeat ties for all developers who work on the repository by the total number of pairs that exist in a repository to compute a measure of repeat ties for the repository.

Control variables

Developer Direct Ties: For each project member, we counted the number of ties that the project member has with developers other than focal project members.

Repository Direct Ties: We take an average of this number over all repository members to compute a measure of direct ties for the repository.

Developer Indirect Ties: A frequency decay measure for indirect ties. First, for each project member, we could count the number of developers with whom the member does not have a direct tie but can reach through others. Next, we account for a weakening of tie strength as the distance between two developers increases.

Repository Indirect Ties: We divided the previous measure by the number of project members to calculate a measure of indirect ties for a project. These measures controls for the capacity of a project to acquire explicit knowledge from outside.

Repository Human capital: Number of developers (i.e., project size) associated with a repository.

Project support: The cumulative number of issues closed in the repository.

Repository interest: The cumulative number of forks, watches, and stars in the repository.

Project Age: The number of months since a first activity in the repository at GitHub.

Alternative variables bank:

* Technological diversity
* External cohesion
* Number of releases
* Dependencies on the project

Estimation method

Hierarchical Bayes Estimation Procedure

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

H3: The nuanced relationship between project structure and popularity: *The degree of superposition, that is, the ratio between the total number of versions of the OSS project to the total number of individual contributions to the project, has a nonlinear relationship with project attractiveness in the OSS community: A moderate degree of project superposition is better for a project’s popularity than very high or very low levels of superposition* (Medappa and Srivastava 2019).

DVs:

Repo popularity: The number of stars

Indep Vs:

Superposition: The ratio of the total number of versions of the OSS repository to the total number of individual task contributions to the repository.

A ~~version~~ release: Each version is a snapshot of the project (repository) at a point in time that sees tasks being added to the project.

The number of versions within a timeframe: Number of differentiated snapshots (versions) within period of time.

To do: find release dates – Done.

The total number of individual task contributions: Number of commits and pull requests made by each developer between 2 ~~versions~~ releases.

Data collection:

Data test: find releases for month <https://docs.github.com/en/webhooks-and-events/events/github-event-types#releaseevent>

Create a table:

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

H4: The importance of a project’s embeddedness among other OSS projects: *A project’s visibility and embeddedness in the global OSS community is positively associated with its long-term sustainability, new developers’ recruitment, a succession of development talent, and diversity of ideas.* (Grewal et al. 2006)

DVs:

Project success: The number of commits for project before date t.

Indep Vs:

Repository Degree centrality

Repository Betweenness centrality

Repository Eigenvector centrality

Data collection:

Create a table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actor.id | Repo\_name\_1 | Repo\_name\_2 | ... | Repo\_name\_n |
| actorid001 |  |  |  |  |
| actorid002 |  |  |  |  |
| ... |  |  |  |  |

Create a table in SQL:

|  |  |  |
| --- | --- | --- |
| Actor.id | Repo\_names | Month |
| actorid001 |  |  |
| actorid002 |  |  |
| ... |  |  |

Query all repos by an actor during the interval of time (e.g., month), record them in field Repo\_names.

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

Summary:

Data collection from BigQuery (at scale)

1. Decide the Time Unit of Analysis, i.e., week, month, etc. – Day
2. Unit of data collection: Repo – day – actor.id
3. Shortlist the list of activities we are calculating:

* num\_dist\_commits
* num\_dist\_commitcomments
* num\_actors\_pushevents
* num\_actors\_pusheventscomment
* num\_dist\_pullreqopened
* num\_dist\_pullreqclosed
* num\_dist\_pullreqAll
* num\_dist\_pullreqcomments
* num\_actors\_pullreq
* num\_actors\_pullreqcomment
* num\_dist\_issuesopened
* num\_dist\_issuesclosed
* num\_dist\_issuesAll
* num\_dist\_issuecomments
* num\_actors\_issues
* num\_actors\_issuescomment
* num\_actors\_allevents
* num\_actors\_issues\_opened
* num\_actors\_issues\_closed
* num\_forks\_event
* num\_actors\_forks
* num\_watch\_event
* num\_actors\_watch

1. Collect data from GH Archive tables 201508-today
   1. Collect and save in .csv separately by years:
      1. Stage I: 2015
      2. Stage II and on: 2016, 2017, 2018, 2019Q1, 2019H2, 2020H1, 2020H2, 2021H1, 2021H2, 2022H1, 2022H2, 2023H1.
   2. Break down in quarters (3 months) if # of records larger than 1,000,000.
   3. It’s going to be a big database!

**Table 1. Database for Ethereum GitHub Project**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Time Unit of Analysis | Repo\_name | Actor.id | Actor\_login | Num\_activities | Num\_activities | ... | Num\_activities |
| Time1 | Repo.name1 | actorid001 |  |  |  |  |  |
| Time1 | Repo.name1 | actorid002 |  |  |  |  |  |
| Time1 | Repo.name1 | ... |  |  |  |  |  |
| Time1 | Repo.name2 | actorid001 |  |  |  |  |  |
| Time1 | Repo.name2 | actorid002 |  |  |  |  |  |
| Time1 | Repo.name2 | ... |  |  |  |  |  |
| ... | ... | ... |  |  |  |  |  |
|  | Repo.nameN | actorid002 |  |  |  |  |  |
|  | Repo.nameN | ... |  |  |  |  |  |

Other interesting analyses that require additional computation may be performed within a shorter timeframe.