OSS Project

The University of Auckland | 825159894

How does the number of issues change over time

Ayato Tanemura

2022

# Introduction

**What is OSS?**

An operations support system (OSS) is a software component that enables a service provider to monitor, control, analyse, and manage the services on its network.

When you have a product that someone is working on because they want to work on it, (not just because they’re getting paid) their personal drive to add creativity and contribute their best ideas tends to be a lot higher. This is often what inspires enthusiast open source communities to develop features that are new and disruptive, and why we see open source projects leading the way in terms of this disruption.

During the last decade, the community-based software development such as OSS has emerged as a new paradigm. The most successful OSS projects such as operating systems-Linux, server application-Apache, desktop environment-Gnome, data analytic packages-R, STATA, programming languages Perl, Python, and database- MySQL have created significant impact [1] on the paradigm of software development. According to the open-source portal SourceForge, more than 58% of the OSS project fail and do not move beyond alpha development stage [2]. In the present scenario of IT world, many firms are adopting or incorporating OSS into their projects. The idea of open source projects is getting more popular because anyone can use the OSS projects codes and customize them according to their requirements. In recent times, the private organizations and government institutions are giving importance to OSS adoption and compare the cost of development between OSS and closed proprietary software development. Adopting OSS has become an integral part of the firm’s strategy. In proprietary software, the level of advancement of that particular project is limited to the vision of those few people. But in the case of open source projects, the project is available to everyone and hosting services like GitHub, OpenSource, and SourceForge enable them. These firms provide services beyond just hosting and providing the version control of the software. The case study [4] shows the way to on-boarding new members of the firm into the virtual OSS development team. The delivery of the software services over the internet and the cloud computing, the reduced cost of the hardware, the improved speed of the digital services play important role in making OSS a popular option. They are able to bring a new outlook and innovation to the existing mechanism which seems to be superior to the proprietary software.

GitHub is usually the most popular site that hosts a large number of open source projects. The top contributors of GitHub are Microsoft, Google, Facebook etc., who consider that the open source projects can bring new features, functionality, and benefits. Each GitHub repository has a feature known as ‘Fork’ that suggests how many times the project has been copied by others. Other characteristics such as the number of stars, the number of pull requests, and the number of subscribers also used to measure the success of OSS projects.

The above-mentioned features or parameters would provide an idea of how many such projects are there on GitHub which can be considered as successful projects. This study investigates how the success of a project is linked to the parameters such fork count, watchers as well as the number of pull requests of the project.

# Research Question and Hypothesis

There are several theories and tools used for analysing the OSS phenomenon. The social network theory helps to model the interaction between different players in the OSS projects. It models actors as the nodes and their relationships as the edges of the graph. The collaborative networks is a variation of the social networks and it can be modelled used the social network theory.

Do the trajectories of the number of issues differ by the number of members, commits, watchers, and pull requests?

1. Issues count as a measure of OSS project performance

An Issue is a note on a repository about something that needs attention. It could be a bug, a feature request, a question or lots of other things. On GitHub you can label, search and assign Issues, making managing an active project easier. These issues include not only problems and bugs but also requests of better solutions or recommendations. Hence, we can say that more issues, better performance in each project.

1. The total number of Members

Members shows the number of members in a project. If the number of members is huge, then we can say two theories. One is that there are many perspectives in the project so the quality increases. On the other hand, if there are too many, there would be conflict within the team members so the quality might decrease. Although I cannot say either positive or negative effect, there might be an effect by the number of members.

H1: The total number of members has an effect on the OSS project performance over time.

1. The total number of Watchers

GitHub provides social features that allow for a community of contributors to be built around the codebase. Any user can “watch” a public repository to receive notifications about events in a repository such as new commits, pull requests, and issues. “Watching” a repository signals interest in the repository’s activity and a potential interest in contributing. It can be seen as a passive type of project membership. If the number of watchers increase, there are more perspectives from the outside of project team. There would be more opportunities to fine issues.

H2: The total number of watchers has a positive effect on the OSS project performance over time.

1. The total number of Commits

A fundamental unit of work in programming is the code contribution (“commit”) that a developer makes to the code base of the project in work. A commit, or "revision", is an individual change to a file (or set of files). When you make a commit to save your work, Git creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of the specific changes committed along with who made them and when. Commits usually contain a commit message which is a brief description of what changes were made. If the number of commits increases, the number of points that might cause issues increase. Hence, we can say more commits, more issues.

H3: The total number of commits has a positive effect on the OSS project performance over time.

1. The total number of Pull Requests

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

The process for a pull request approval in Git will involve getting the project maintainer(s) to review your work; after which they will provide comments or, if your pull request is approved, will merge your changes directly into the main repository.

If the number of pull request increase, the number of reviewing opportunities increase. Hence the chance of issues found increase as well.

H4: The total number of pull requests has a positive effect on the OSS project performance over time.

# Definition of main variables, visualisation exploration

## Definition of main variables

1. Issues

The total numbers of issues is the dependent variable of this research project which is considered as a proxy for the OSS project quality. The issues count is measured from the issues requests for new features from the GitHub repository for each project. Whenever any developer raised or requested issues to the OSS project repository, its issues count increases.

1. The number of Members

Members shows the number of members who assigned in a project team. Whenever anyone joined in the project, its members count increase.

1. The number of Commits

This gives the number of commits to each project. This is collected from all branches in the project. Whenever anyone in the team members change edit the project, its commits count increase.

1. The number of Watchers

Watchers provide the number of users who opted to watch the progress of the project and to get the notifications regarding the development. The value of this variable is obtained through watchers parameter of the GitHub repository for each OSS project.

1. The number of Pull Request

The pull request count is measured from the pull requests for new features from other users of the GitHub repository for each project. Whenever anyone created pull request to the OSS project repository, its pull request count increases.

1. Time

Dataset used in this project has two years of record. The data is recorded in each quarter and thus there are totally eight quarters frames. 1 shows the origin point of each project and 8 is the last recorded point.

## Visualisation exploration

1. Sampling visualisation (Appendix A)

This is the empirical growth plots with superimposed OLS trajectories for 5 participants in the OSS project research. This is identifying a suitable functional form for the level-1 sub-model. As you can see the Appendix A, Issues and Time appears linear between 1st quarter and 8th quarter. For project ID 3671, 3721, and 5378, Issues tend to increase gradually from 1st to 8th quarter. For project ID 3085, Issues increase significantly from 1st to 8th quarter. However, for 2647, there is no significant change in over two years. Generally, there is considerable changes in Issues over years. And it is a linear trajectory for the identified change.

1. Each predictor’s visualisation

**Appendix B**

The graph shown in Appendix B represents the total number of issues by the total number of members in projects. As members increase, the total number of issues increase very slightly. The relationship appears to unstable because it repeats increase and decrease. To model Members, it seems like there is no effect on rate of change over time, hence, no need to interact with time variant.

**Appendix C**

The graph shown in Appendix C represents the total numbers of issues by the total number of commits. As commits increase, the total number of issues increase. The relationship appears to curve very slightly because it is getting slightly flatten out for higher commits. To model this predictor, I need to test with both original value and a square term to justify the better value.

**Appendix D**

The graph shown in Appendix D represents the total number of issues by the total number of watchers. It can be seen the similar as Appendix B which repeats increase and decrease over time. To model Watchers, it seems like there is no effect on rate of change over time, hence, no need to interact with time variant.

**Appendix E**

The graph shown in Appendix E represents the total number of issues by the total number of pull requests. As pullReq increase, the Issues also tend to increase. The relationship appears to curve slightly because it flattens out for higher pullReq values. To model the curvature, the analysis include a squared term in the model.

# Data cleaning and preparation

For the preparation of data, I created Time\_1 column to add another time factor. The Time column in original data shows a sequence for time of observation with starting from 1 to 8. However, in my case study, I would like to set the start point as 0. Thus, I created the Time\_1 column to show the sequence for time observation starting with 0. Other than the Time factor, I used Members, Commits, Issues, Watchers, and PullReq for the predictors which were continuous variables without issues.

In terms of pull request predictor, as you can see in the Appendix E, the response variables are distributed exponentially. Hence, I used log transformation theory to make is as normal as possible so that the statistical analysis results from this data become more valid. The log transformation reduces or removes the skewness of the original data. The importance caveat here is that the original data has to follow or approximately a long normal distribution.

# Methodology

In the analysis phase, I first estimated the full model and then drop the insignificant terms of the models. I applied stepwise regression method to finalise the model. Stepwise regression is the step-by-step iterative construction of a regression model that involves the selection of independent variables to be used in a final model. It involves adding or removing potential explanatory variables in succession and testing for statistical significance after each iteration. In this report I will show the final and the penultimate models. The penultimate model is Model A, and the final is Model B.

Model A includes Commits, Watchers, and Log(pullReq) as a predictor of both initial status and rate of change, but Members as a predictor of only initial status. In Model B, it includes Commits and Watchers as a predictor of both initial status and rate of change, but Members and Log(pullReq) as predictors of only initial status.

Results of both models are shown in Table 1. P-value of the parameter estimates are reported in parentheses. Composite model, model interpretation, and variance components for each model is mentioned in the following section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 1 | | | | |
|  | Model A | | Model B | |
| **Fixed effects** | Value | p-value | Value | p-value |
| Intercept | -1.958 | (0.1464) | -2.16 | (0.1076) |
| Members | 0.192 | (0.0328) | 0.187 | (0.0377) |
| Commits | 0.0347 | (0.0000) | 0.343 | (0.0000) |
| Time\_1 | 0.511 | (0.5635) | 0.718 | (0.4130) |
| Watchers | 0.00596 | (0.1634) | 0.00576 | (0.1770) |
| Log (pullReq + 1) | 7.816 | (0.0000) | 8.185 | (0.0000) |
| Commits \* Time\_1 | 0.00989 | (0.0000) | 0.0101 | (0.0000) |
| Time\_1: watchers | 0.00907 | (0.0000) | 0.00931 | (0.0000) |
| Time\_1: Log (pullReq + 1) | 0.247 | (0.2027) |  |  |
| **Variance Components** |  |  |  |  |
| Within-person | 135.326 |  | 135.129 |  |
| Between-Person |  |  |  |  |
| Initial status | 268.578 |  | 268.749 |  |
| Rate of change | 240.225 |  | 243.717 |  |
| ICC | 0.528 |  | 0.524 |  |
| **Goodness-of-fit** |  |  |  |  |
| AIC (smaller is better) | 23221.12 |  | 23220.7 |  |
| BIC (smaller is better) | 23297.74 |  | 23291.43 |  |

## Model justification

Model A: Members (only initial status), Commits, Watchers, and Log (Pull Requests)

**Model Equation**

Level-1 Model: Issues = X\_0i + X\_1i \* Time\_1 + j

Level-2 Model:

X\_0i = - 1.958 + 0.192 Members + 0.0347 Commits + 0.00596 Watchers + 7.816 Log(pullReq) + y\_0i

X\_1i = 0.511 + 0.00989 Commits + 0.00907 Watchers+ 0.247 Log(pullReq) + y\_1i

Composite model:

**Model Interpretation**

1. The estimated initial Issues controlling for members, commits, watchers, and pull requests is -1.96 (p > 0.05). However, it is non-significant at 0.05 level of significance.

2. The estimated differential in initial Issues for one member difference in Members controlling for other predictors at the initial stage is 0.19 (p<0.05) at 0.05 level of significance.

3. The estimated differential in initial Issues for one commit difference in Commits controlling for other predictors at the initial stage is 0.035 (p<0.01) at 0.01 level of significance.

4. The estimate rate of change in Issues controlling for Members, Commits, Watchers, and Log (Pull Requests) is 0.51 (p>0.05). However, it is non-significant at 0.05 level of significance.

5. The estimated differential in initial Issues for one watcher difference in Watchers controlling for other predictors at the initial stage is 0.0060 (p>0.05). However, it is non-significant at 0.05 level of significance.

6. The estimated differential in initial Issues for one pull request difference in Log (Pull Request) controlling for other predictors at the initial stage is 7.82 (p<0.01) at 0.01 level of significance.

7. The estimated differential in the rate of change in Issues of Commits is 0.0099 (p < 0.01) at 0.01 level of significance.

8. The estimated differential in the rate of change in Issues of Watchers is 0.0091 (p < 0.01) at 0.01 level of significance.

9. The estimated differential in the rate of change in Issues of Pull Request is 0.25 (p>0.05). However, it is non-significant at 0.05 level of significance.

**Variance Components**

Level 1 (within person variance) gets the estimate of 135.326

Level 2 (between person variance) receives the estimate of

268.578 for the initial status and

240.225 for the rate of change

ICC = 0.528 means 52.8% variation in employment rate is attributable to differences among projects.

Model B: Members (only initial status), Commits, Watchers, and Log (Pull Requests) (Only initial status)

**Model Equation**

Level-1 Model: Issues = X\_0i + X\_1i \* Time\_1 + j

Level-2 Model:

X\_0i = - 2.16 + 0.187 Members + 0.0343 Commits + 0.00576 Watchers + 8.185 Log(pullReq) + y\_0i

X\_1i = 0.718 + 0.00101 Commits + 0.00931 Watchers + y\_1i

Composite model:

**Model Interpretation**

1. The estimated initial Issues controlling for members, commits, watchers, and pull requests is -2.16 (p > 0.05). However, it is non-significant at 0.05 level of significance.

2. The estimated differential in initial Issues for one member difference in Members controlling for other predictors at the initial stage is 0.19 (p<0.05) at 0.05 level of significance.

3. The estimated differential in initial Issues for one commit difference in Commits controlling for other predictors at the initial stage is 0.035 (p<0.01) at 0.01 level of significance.

4. The estimate rate of change in Issues controlling for Members, Commits, Watchers, and Log (Pull Requests) is 0.72 (p>0.05). However, it is non-significant at 0.05 level of significance.

5. The estimated differential in initial Issues for one watcher difference in Watchers controlling for other predictors at the initial stage is 0.0058 (p>0.05). However, it is non-significant at 0.05 level of significance.

6. The estimated differential in initial Issues for one pull request difference in Log (Pull Request) controlling for other predictors at the initial stage is 8.18 (p<0.01) at 0.01 level of significance.

7. The estimated differential in the rate of change in Issues of Commits is 0.010 (p < 0.01) at 0.01 level of significance.

8. The estimated differential in the rate of change in Issues of Watchers is 0.0093 (p < 0.01) at 0.01 level of significance.

**Variance Components**

Level 1 (within person variance) gets the estimate of 135.129

Level 2 (between person variance) receives the estimate of

268.749 for the initial status and

243.717 for the rate of change

ICC = 0.524 means 52.4% variation in employment rate is attributable to differences among projects.

## Model selection

Our results are robust, because we see that parameter estimates do not change from Model A to Model B. To test the goodness of the fit of our models, we report deviance, AIC, and BIC statistics. Lower values of AIC and BIC statistics mean a better fit for the model. According to Table 1, both AIC and BIC have smaller values in Model B. Therefore, I will use Model B as a final model.

# Results and discussions

## Results

My research approached to the trajectories of the number of issues differ by the number of members, commits, watchers, and pull requests. Based on the data analysis, this study found that the variables number of members, commits, watchers, and pull requests show positive influence on the performance of the project in initial status. Also, the variables number of commits and watchers show the significant positive influence in rate of change over time. The result support the hypothesis H2 and H3. The number of members and pull requests support the initial status, however, there is no effect over the period.

|  |  |
| --- | --- |
| Hypothesis | Result |
| H1: The total number of members has a positive effect on the OSS project quality over time. | Not Supported |
| H2: The total number of commits has a positive effect on the OSS project quality over time. | Supported |
| H3: The total number of watchers has a positive effect on the OSS project quality over time. | Supported |
| H4: The total number of pull requests has a positive effect on the OSS project quality over time. | Not Supported |

## Discussion

The results will be useful for future projects. Since the points made by non-project members are from a different perspective, new ideas and values can be gleaned from them. This result has given us an idea of how we can increase the number of Issues. For example, among these four predictors, it is effective to increase the number of Pull Requests. The next question is how to increase the number of Pull Requests. The important point is how to produce one result and how to make use of that result.

# Conclusion

This research paper approached the performance of OSS project. The performance can be measured by the number of issues in the projects. I applied longitudinal analysis to interpret how the number of issues changes over two years and if there was any interactions from four other predictors; members, commits, watchers, and pull requests. I used two models to justify the model quality and accuracy of predictions. As a result, all predictors have significant effects on initial status, however, only commits and watchers have significant effects on rate of change.

This study can be extended by collecting a cross sectional data on OSS projects with some additional variables, for example, programming language type, date of using OSS, and degree, etc..

# Appendix

Diagram

Description automatically generated with low confidenceAppendix A

Appendix B

Chart, line chart, scatter chart

Description automatically generated

Appendix C

Chart, scatter chart

Description automatically generated

Appendix D

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generatedAppendix E

Appendix F

# References