An Empirical Analysis of Mozilla Bug Data

by Mohemmed Faisal, Kartikey Bhalla, Preeti Pothireddy, Simran Thakur, Aparna V

Msc Business Analytics, Bayes Business School

Chapter One: Cleaning, Manipulating and Structuring of Data:

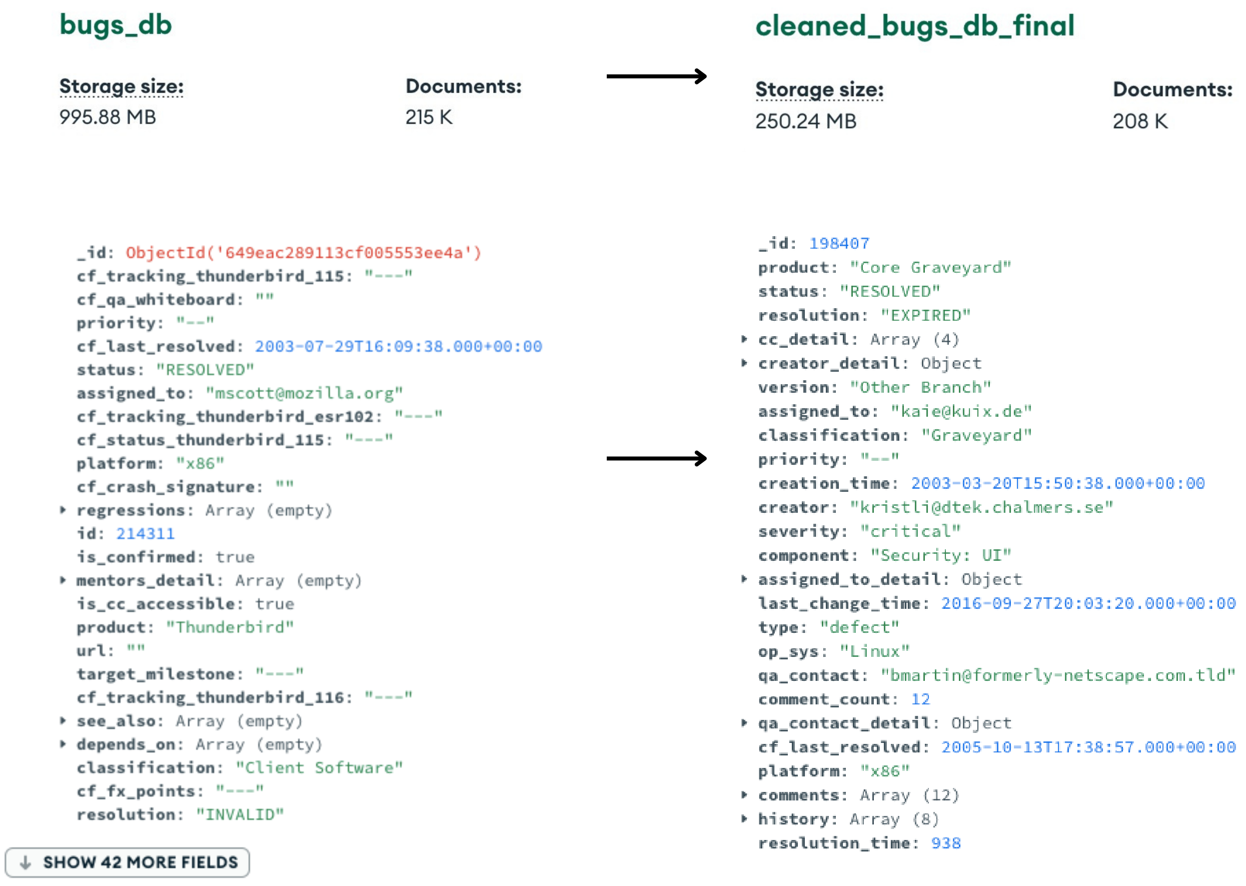
Mozilla is an open source software originating from Netscape, a computer services company. Mozilla extensively uses Bugzilla, a bug tracking system, to store, track and manage bugs for various projects. For the purpose of our research, we handled a real word dataset, comprising 2,15,173 bugs data for the Mozilla project hosted at Bugzilla, between a development window of 1997 to 2003. As the preliminary step of our analysis, we cleaned, manipulated and restructured the bug data, using MongoDB in the following ways:

- Only bugs which were officially closed have been considered (i.e field '*is_open'* = false).
- Redundant fields such as *'assigned_to'*, *'qa_contact'* and *'creator'* have been removed, as the same information was also made available within nested objects.
- All the fields with prefix *'cf'* have been removed, except for '*cf_last_resolved*', since the fields were inconsistent across the documents and did not add much significant value.
- 139 bugs which were closed but did not have a resolution time have been dropped.
- Unnecessary nested fields in the object *comments* (except '*id*', '*comment_creator*' and '*creation_time*') , which did not align with our analysis, have been removed. *Comment text* has been dropped as we did not intend to perform natural language processing.
- We identified bug resolution time as a key metric for our further analysis, and hence a new field '*resolution_time*' was created by subtracting '*creation time*' from '*cf_last_resolved*'.

Overall, the aforementioned steps of cleaning, manipulating and restructuring of the bug data offered tri-fold benefits:

- Ensured the bug data was organised in a consistent, coherent and readable format. Maintaining a consistent data structure enabled easier querying, aggregating, and analysis of our bug data in MongoDB

- Significantly reduced the data volume by approximately 75%, from 995.88 MB to 250.24 MB. The number of documents were reduced to 207,563 and the number of fields were reduced to 24. Reduction in big data at the earliest enhances the data management and data quality and therefore improves the operations of big data systems (Rehman et al., 2016).
- Made it much easier to perform meaningful analysis and extract valuable insights. Clean and well-structured data helped efficiently identify trends, patterns and correlations in the bug data, which will be discussed later on in this report



bugs_db

Storage size: 995.88 MB
Documents: 215 K

cleaned_bugs_db_final

Storage size: 250.24 MB
Documents: 208 K

```
  _id: ObjectId('649eac289113cf005553ee4a')
  cf_tracking_thunderbird_115: "---"
  cf_qa_whiteboard: ""
  priority: "--"
  cf_last_resolved: 2003-07-29T16:09:38.000+00:00
  status: "RESOLVED"
  assigned_to: "mscott@mozilla.org"
  cf_tracking_thunderbird_esr102: "---"
  cf_status_thunderbird_115: "---"
  platform: "x86"
  cf_crash_signature: ""
▸ regressions: Array (empty)
  id: 214311
  is_confirmed: true
▸ mentors_detail: Array (empty)
  is_cc_accessible: true
  product: "Thunderbird"
  url: ""
  target_milestone: "---"
  cf_tracking_thunderbird_116: "---"
▸ see_also: Array (empty)
▸ depends_on: Array (empty)
  classification: "Client Software"
  cf_fx_points: "---"
  resolution: "INVALID"
```

```
  _id: 198407
  product: "Core Graveyard"
  status: "RESOLVED"
  resolution: "EXPIRED"
▸ cc_detail: Array (4)
▸ creator_detail: Object
  version: "Other Branch"
  assigned_to: "kaie@kuix.de"
  classification: "Graveyard"
  priority: "--"
  creation_time: 2003-03-20T15:50:38.000+00:00
  creator: "kristli@dtek.chalmers.se"
  severity: "critical"
  component: "Security: UI"
▸ assigned_to_detail: Object
  last_change_time: 2016-09-27T20:03:20.000+00:00
  type: "defect"
  op_sys: "Linux"
  qa_contact: "bmartin@formerly-netscape.com.tld"
  comment_count: 12
▸ qa_contact_detail: Object
  cf_last_resolved: 2005-10-13T17:38:57.000+00:00
  platform: "x86"
▸ comments: Array (12)
▸ history: Array (8)
  resolution_time: 938
```

Chapter Two: Descriptive Insights:

Insight 1:

Figure 1

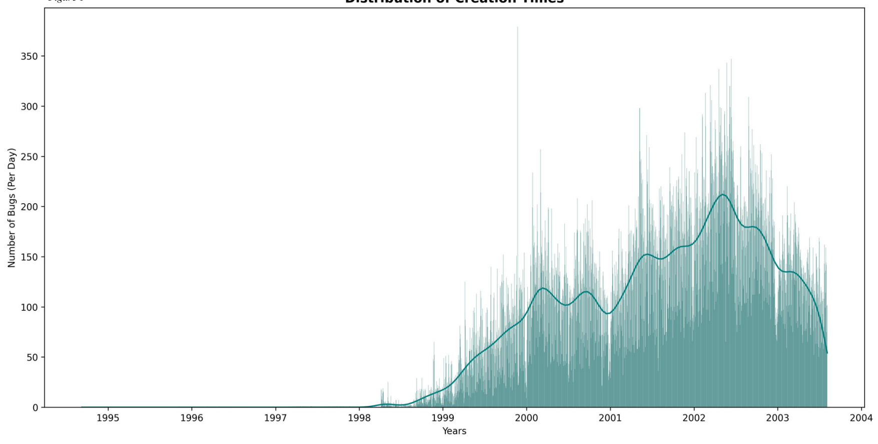
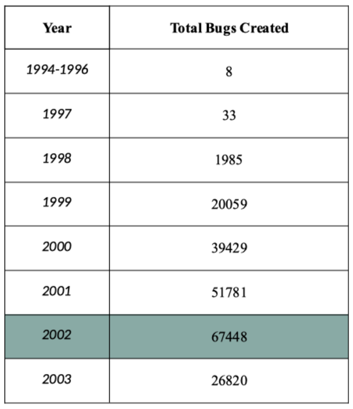


Table 1

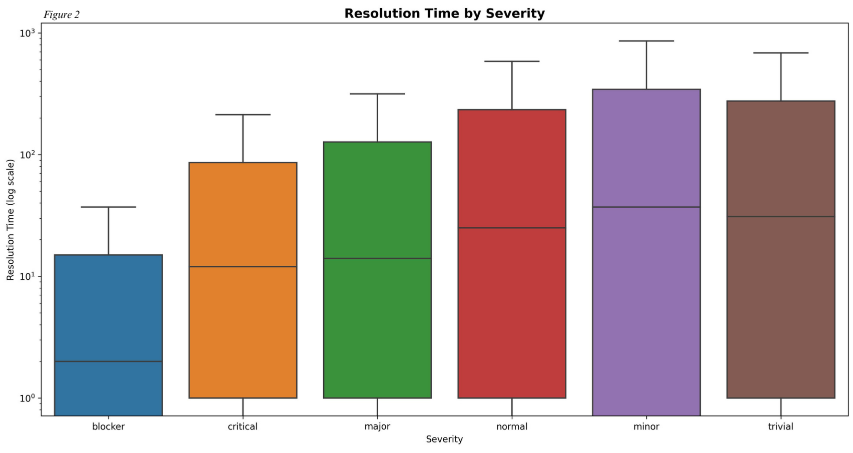


Year	Total Bugs Created
1994-1996	8
1997	33
1998	1985
1999	20059
2000	39429
2001	51781
2002	67448
2003	26820

Figure 1 highlights the distribution of bug creation time and table 1 highlights the total number of bugs created each year. Clearly, in 2002, we can notice a significant jump in the number of bugs being created. Our research identified four notable releases in the Mozilla project during 2002 that had a possible association with the higher bug reports:

- Mozilla 1.0 Release (June 5, 2002): This was a significant milestone for the project, that aimed to provide an all-inclusive set of web browsing and email capabilities.
- Mozilla 1.1 Alpha Release (October 2, 2002): Early and unstable version of the software, intended for testing and development purposes (alpha releases contain several bugs and the focus is on finding and fixing critical bugs).
- Mozilla 1.2 Beta Release (November 1, 2002): Pre-release version of the Mozilla web browser and suite (beta releases aim to identify bugs/issues that need to be addressed before the final stable release).
- Mozilla 1.2 Release (November 26, 2002): Significant stable release of the Mozilla web browser and suite, following the alpha and beta releases.

Insight 2:



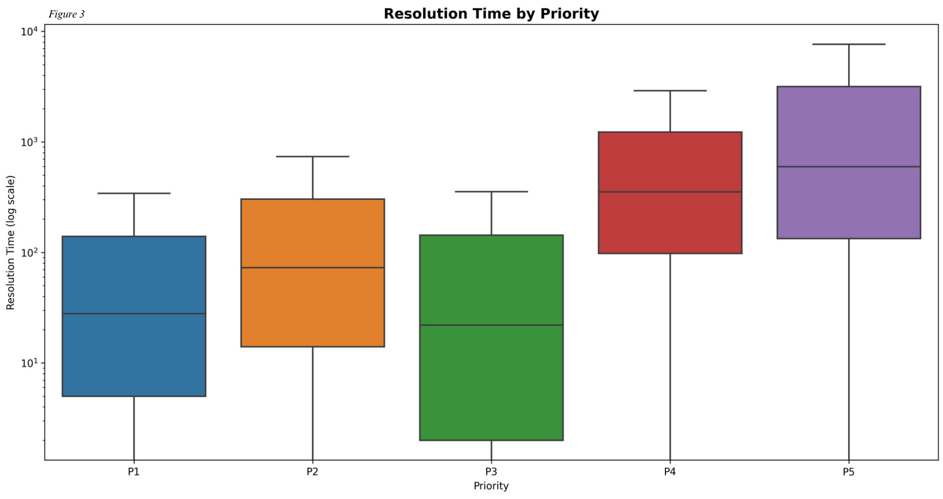


Figure 2 and Figure 3 illustrate bug resolution time (represented on a log scale due to high variation) against two bug attributes:

(a) Severity (i.e. how serious is the bug: blocker > critical > major > normal > minor > trivial)
(b) Priority (i.e. how important is the bug: P1 > P2 > P3 > P4 > P5)

Evidently, the median resolution time increases with decreasing levels of severity and priority i.e high priority bugs and high severity bugs are resolved quicker than low priority bugs and low severity bugs. While this makes logical sense, interestingly, bugs classified as *trivial*, although less severe than *minor* bugs, are resolved faster. Similarly, *P3* bugs, although less in priority than *P1* and *P2* bugs, are also resolved faster. This might be because they are rather simple to fix (Nguyen et al, 2014).

Insight 3:

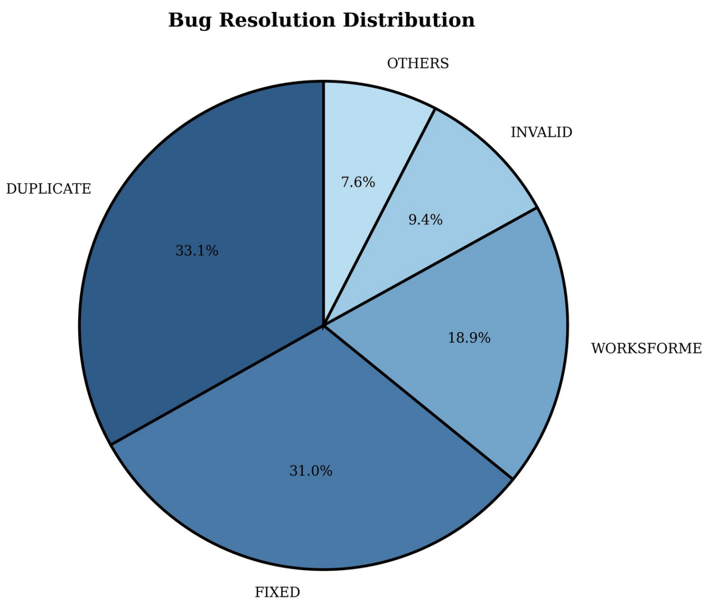


Figure 4

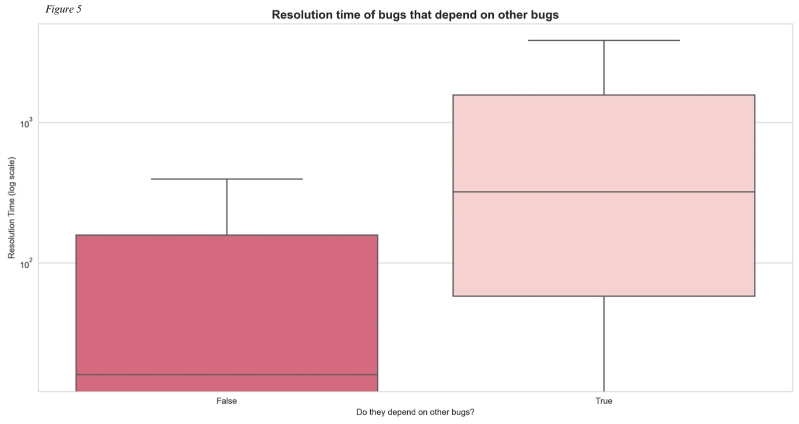
Table 2 **Bug Statistics for Products**

Product Name	Total Bugs	Total Duplicates	'%' of duplicates
SeaMonkey	67653	29145	43.08%
Firefox	2932	1161	39.6%
MailNews Core	16492	5896	35.75%
Camino Graveyard	3030	995	32.84%
Core	73679	22578	30.64%
Core Graveyard	19840	5429	27.36%
Bugzilla	5043	1272	25.22%
Calendar	1076	216	20.07%
Tech Evangelism Graveyard	6073	688	11.33%
NSS	1530	135	8.82%

Figure 4 reveals the distribution of the bug resolution field. Overall, we identified four dominant labels: (a) Duplicate (b) Fixed (c) Works for me and (d) Invalid. Surprisingly, over 33% of the total bugs have been identified as duplicates. This is consistent with findings from previous research, according to which on average, 32.8%–41.6% of bug reports remain duplicates of one another in bug tracking systems. This can be attributed to the asynchronous nature of bug report submission, because of which traditional bug tracking systems such as Bugzilla cannot prevent duplicate bug reports (Jahan, S et al). On further investigation, it came to our notice that over 43% of SeaMonkey bugs and approximately 40% of Firefox bugs are duplicates, which can be a point of concern (Table 2). The above findings highlight the criticality of duplicate bug reporting in Mozilla and warrant further investigation.

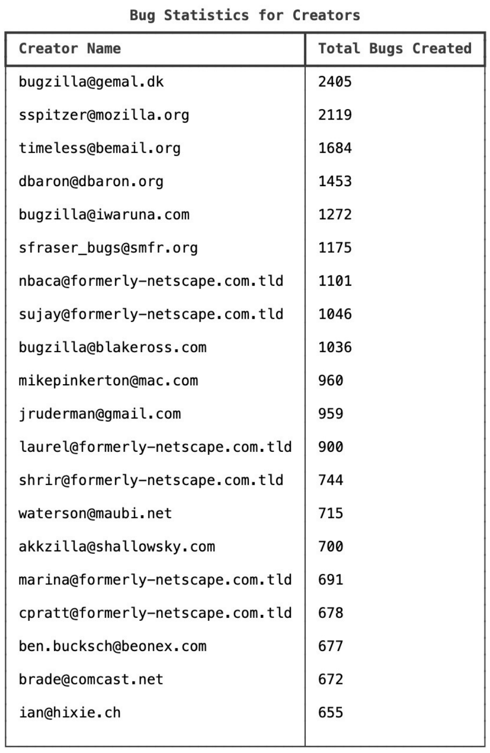
Insight 4:

Figure 5



In Mozilla's Bugzilla, the *depends_on* field represents dependencies between two bugs. In other words, it is used to indicate that one bug (i.e. the dependent bug) relies on another bug (i.e. the blocking bug) to be resolved or fixed before it can be addressed. Intuitively, the dependent bug should take longer to resolve. The same is iterated from figure 5, according to which the median resolution time of the dependent bug ("*depends_on*" = True) is significantly higher than that of the blocking bug ("*depends_on*" = False).

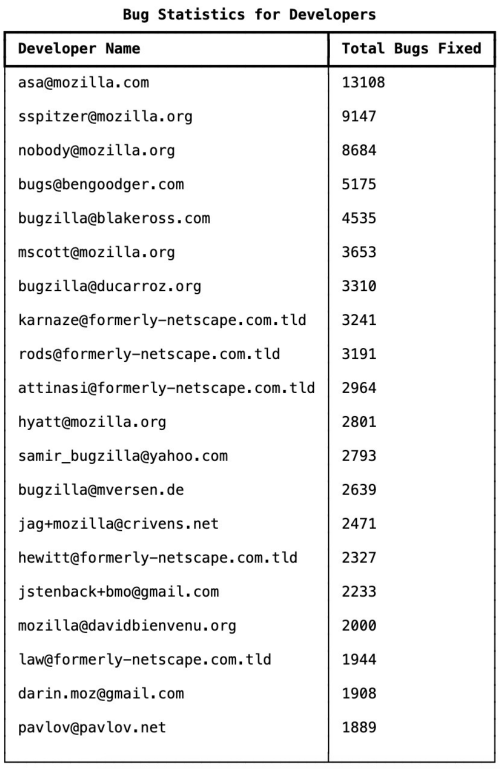
Insight 5:



Bug Statistics for Creators

Creator Name	Total Bugs Created
bugzilla@gemal.dk	2405
sspitzer@mozilla.org	2119
timeless@bemail.org	1684
dbaron@dbaron.org	1453
bugzilla@iwaruna.com	1272
sfraser_bugs@smfr.org	1175
nbaca@formerly-netscape.com.tld	1101
sujay@formerly-netscape.com.tld	1046
bugzilla@blakeross.com	1036
mikepinkerton@mac.com	960
jruderman@gmail.com	959
laurel@formerly-netscape.com.tld	900
shrir@formerly-netscape.com.tld	744
waterson@maubi.net	715
akkzilla@shallowsky.com	700
marina@formerly-netscape.com.tld	691
cpratt@formerly-netscape.com.tld	678
ben.bucksch@beonex.com	677
brade@comcast.net	672
ian@hixie.ch	655

Table 3



Bug Statistics for Developers

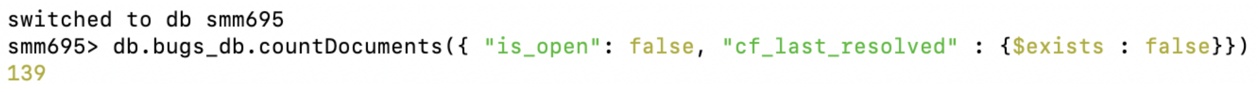
Developer Name	Total Bugs Fixed
asa@mozilla.com	13108
sspitzer@mozilla.org	9147
nobody@mozilla.org	8684
bugs@bengoodger.com	5175
bugzilla@blakeross.com	4535
mscott@mozilla.org	3653
bugzilla@ducarroz.org	3310
karnaze@formerly-netscape.com.tld	3241
rods@formerly-netscape.com.tld	3191
attinasi@formerly-netscape.com.tld	2964
hyatt@mozilla.org	2801
samir_bugzilla@yahoo.com	2793
bugzilla@mversen.de	2639
jag+mozilla@crivens.net	2471
hewitt@formerly-netscape.com.tld	2327
jstenback+bmo@gmail.com	2233
mozilla@davidbienvenu.org	2000
law@formerly-netscape.com.tld	1944
darin.moz@gmail.com	1908
pavlov@pavlov.net	1889

Table 4

Table 3 and Table 4 display the top 20 creators (ranked by the total number of bugs reported) and top 20 developers (ranked by the total number of bugs fixed) respectively. Firstly, some key creators (i.e. bugzilla@gemal.dk and sspitzer@mozilla.org) and key developers (i.e. asa@mozilla.com and sspitzer@mozilla.org) can be identified in the Mozilla ecosystem, whose contributions are significantly higher as compared to the rest. Secondly, bugzilla@blakeross.com and sspitzer@mozilla.org appear in the top 10 positions across both the tables. Hence, they have made substantial contributions both as developers and creators, and could potentially be identified as highly active and influential members of the Mozilla bug tracking community. Thirdly, nobody@mozilla.org is a rank three developer, and on further investigation, it came to our notice that the bugs assigned to this name are open for anyone to take up. Their average resolution time (calculated in the code) is over 3315 days, since large software projects like Mozilla can have substantial backlog of bugs to address, and it may take time for developers to review and take ownership of them. Finally, even though the project is an open source software, majority of the top developers are associated with Netscape or Mozilla.

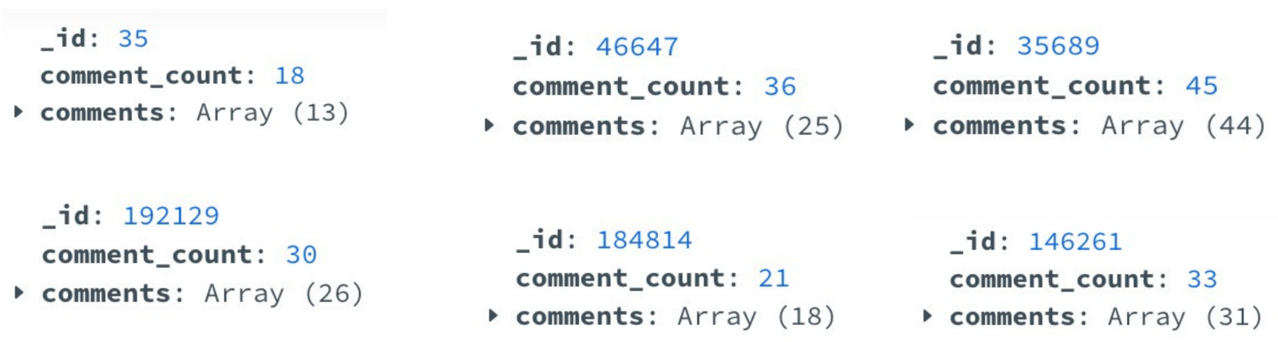
Insight 6:

Figure 8:



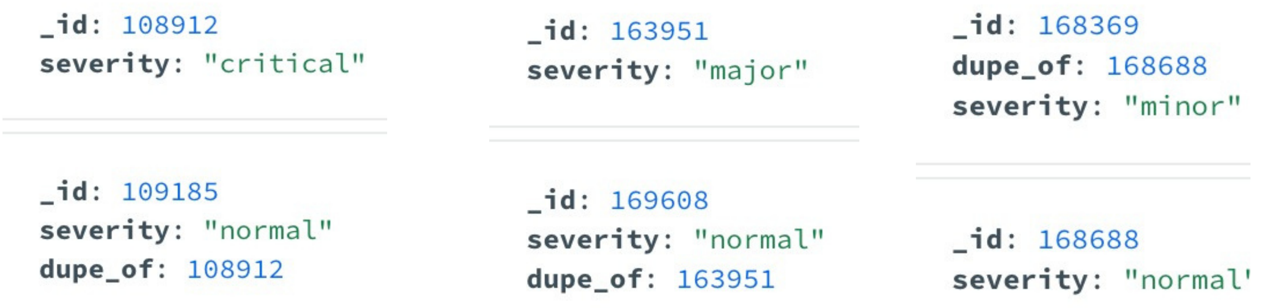
```
switched to db smm695
smm695> db.bugs_db.countDocuments({ "is_open": false, "cf_last_resolved" : {$exists : false}})
139
```

Figure 9:



```
_id: 35
comment_count: 18
▸ comments: Array (13)

_id: 46647
comment_count: 36
▸ comments: Array (25)

_id: 35689
comment_count: 45
▸ comments: Array (44)

_id: 192129
comment_count: 30
▸ comments: Array (26)

_id: 184814
comment_count: 21
▸ comments: Array (18)

_id: 146261
comment_count: 33
▸ comments: Array (31)
```

Figure 10:



```
_id: 108912
severity: "critical"

_id: 163951
severity: "major"

_id: 168369
dupe_of: 168688
severity: "minor"

_id: 109185
severity: "normal"
dupe_of: 108912

_id: 169608
severity: "normal"
dupe_of: 163951

_id: 168688
severity: "normal'
```

The above figures highlight certain inconsistencies in the bug reports:

- Figure 8: There are 139 bugs that have missing *cf_last_resolved* values, even though the bugs were *closed*. Does this mean some users are not following the standard bug resolution process? Were the bugs directly closed without being marked as resolved first?...
- Figure 9: There are 29 documents in the bug dataset wherein *comment_count* is not the same as the number of actual comments in the comment array. Now, are certain comments kept private?...
- Figure 10: On comparing the duplicate bug reports with the original, there exists a mismatch in the severity labels in 30% of the cases (calculated in code). Now, are these labels human-assigned or automatically assigned? If duplicate bug reports have such unreliable severity labels, is it a warning signal on the reliability of the full bug severity data? (Tian et al, 2015)

Overall, certain discrepancies exist in the bug data and additional information is required for further investigation.

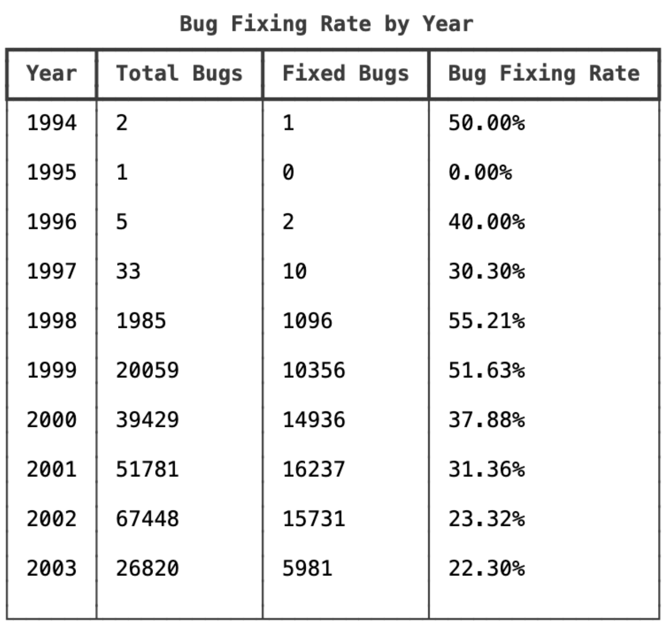
Chapter Three: Insightful Data Analysis:

Part A:

For Part A of our data analysis, we narrowed down our focus on one key metric: Bug Fixing Rate. During a bug lifecycle, a software bug undergoes a series of processes, from identification to closure. When a bug is newly identified, reported and confirmed by developers, the bug status is changed to "Open". Once the triager assigns the bug to a relevant developer, the status of the bug is changed to "Assigned", and once the developer resolves the bug, the status is further changed to either: "Closed", "Resolved" or "Verified", along with a resolution item attached. As discussed earlier, closed bugs can have multiple labels such as: "Duplicate", "Fixed", "Works for me", "Invalid", "Expired" etc. When a closed bug is marked with a "Fixed" resolution, it indicates that the bug reported has been successfully addressed by the developer. Hence, bug fixing rate has been calculated as the ratio of fixed bugs to closed bugs. For instance, in the year *Y*, a creator reported *m* bugs of which *n* bugs are closed. Among the *n* closed bugs, the number of fixed bugs is *k*, then the creator's bug fixing rate of the year is *k*/*n.* Now that we understand the definition of bug fixing rate, next, we analysed the following three research questions, identified in the research paper, "*An Empirical Study of Bug Fixing Rate*" by Zou et al:

RQ1: What is the overall situation of a project's bug fixing rate?

The primary motivation behind analysing the project's overall bug fixing rate is to gain a broader perspective on the Mozilla project, by gauging the overall number of closed bugs that have been successfully fixed or resolved. For this purpose, we considered all fixed bugs and all closed bugs over the years. Table 5 highlights that the bug fixing rate has seen a significant drop over the years, especially between the development window of 1997 to 2003. Our results are counterintuitive, as ideally, the bug fixing rate is expected to rise as the project matures. In summary, the Mozilla project consists of many non-fixable bugs which can be a major concern.



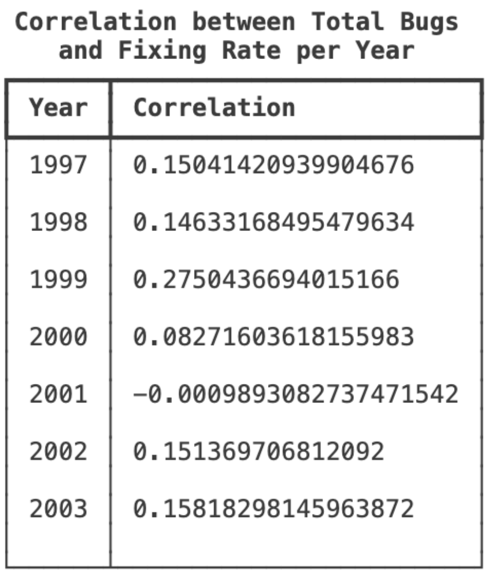
Bug Fixing Rate by Year

Year	Total Bugs	Fixed Bugs	Bug Fixing Rate
1994	2	1	50.00%
1995	1	0	0.00%
1996	5	2	40.00%
1997	33	10	30.30%
1998	1985	1096	55.21%
1999	20059	10356	51.63%
2000	39429	14936	37.88%
2001	51781	16237	31.36%
2002	67448	15731	23.32%
2003	26820	5981	22.30%

Table 5

RQ2: Is there a correlation between the number of bugs reported by a creator and their bug fixing rate?

The primary motivation behind RQ2 is to gain a deeper understanding of the relation between bug reporting and resolution dynamics within the Mozilla project. By examining this relationship, we can understand how a creator's bug reporting behavior influences the efficiency and effectiveness of bug resolution. The idea is that as a creator reports more bugs, they are likely to gain experience in identifying and reporting fixable bugs, which could potentially lead to a higher bug fixing rate. To achieve this, we calculated the correlation coefficient between the two variables: (a) Creator's bug fixing rate and (b) Total number of bugs reported by the creator. Majorly, the correlation between the two variables is positive. In other words, the more a reporter reports, the more likely the bugs he reports will be fixable.



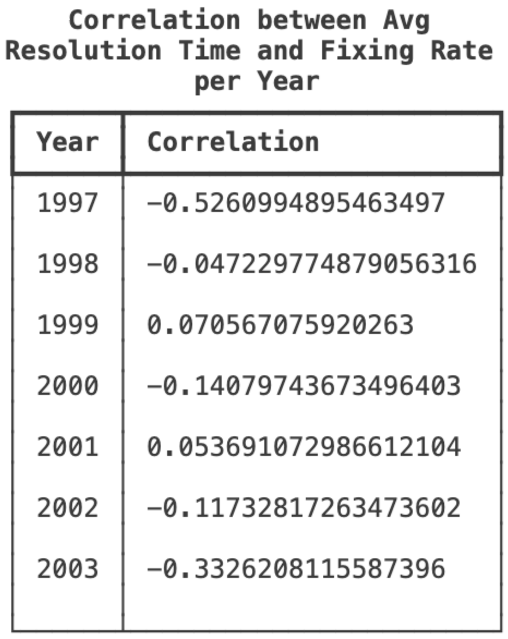
Correlation between Total Bugs and Fixing Rate per Year

Year	Correlation
1997	0.15041420939904676
1998	0.14633168495479634
1999	0.2750436694015166
2000	0.08271603618155983
2001	-0.0009893082737471542
2002	0.151369706812092
2003	0.15818298145963872

Table 6

RQ3: Is it likely that the higher a creator's bug fixing rate is, the less time it takes to close the bugs he reports?

The bug fixing rate of different creators will vary, since some creators may be more experienced at identifying and reporting bugs. Hence, RQ3 aims to explore whether there is a relationship between a creator's bug fixing rate and the time it takes to close their reported bugs. In order to achieve this, we calculated the correlation coefficient between two variables: (a) Creator's bug fixing rate and (b) Average resolution time of the creator per year. Clearly, there exists a significant negative relationship between the two variables (Table 5). In other words, the higher a creator's bug fixing rate is, the less time it takes to close the bugs he reports. Hence, it can be recommended that developers focus more on bugs reported by creators with higher bug fixing rates (Zou et al).



Correlation between Avg Resolution Time and Fixing Rate per Year

Year	Correlation
1997	-0.5260994895463497
1998	-0.047229774879056316
1999	0.070567075920263
2000	-0.14079743673496403
2001	0.053691072986612104
2002	-0.11732817263473602
2003	-0.3326208115587396

Table 7

Part B:

For Part B of our data analysis, we were keen on understanding the network structure of creators and developers in the Mozilla project. We narrowed our focus on the most severe bugs i.e. blocker bugs, and explored two broad questions:

RQ4: What is the overall network structure of creators and developers working on blocker bugs?

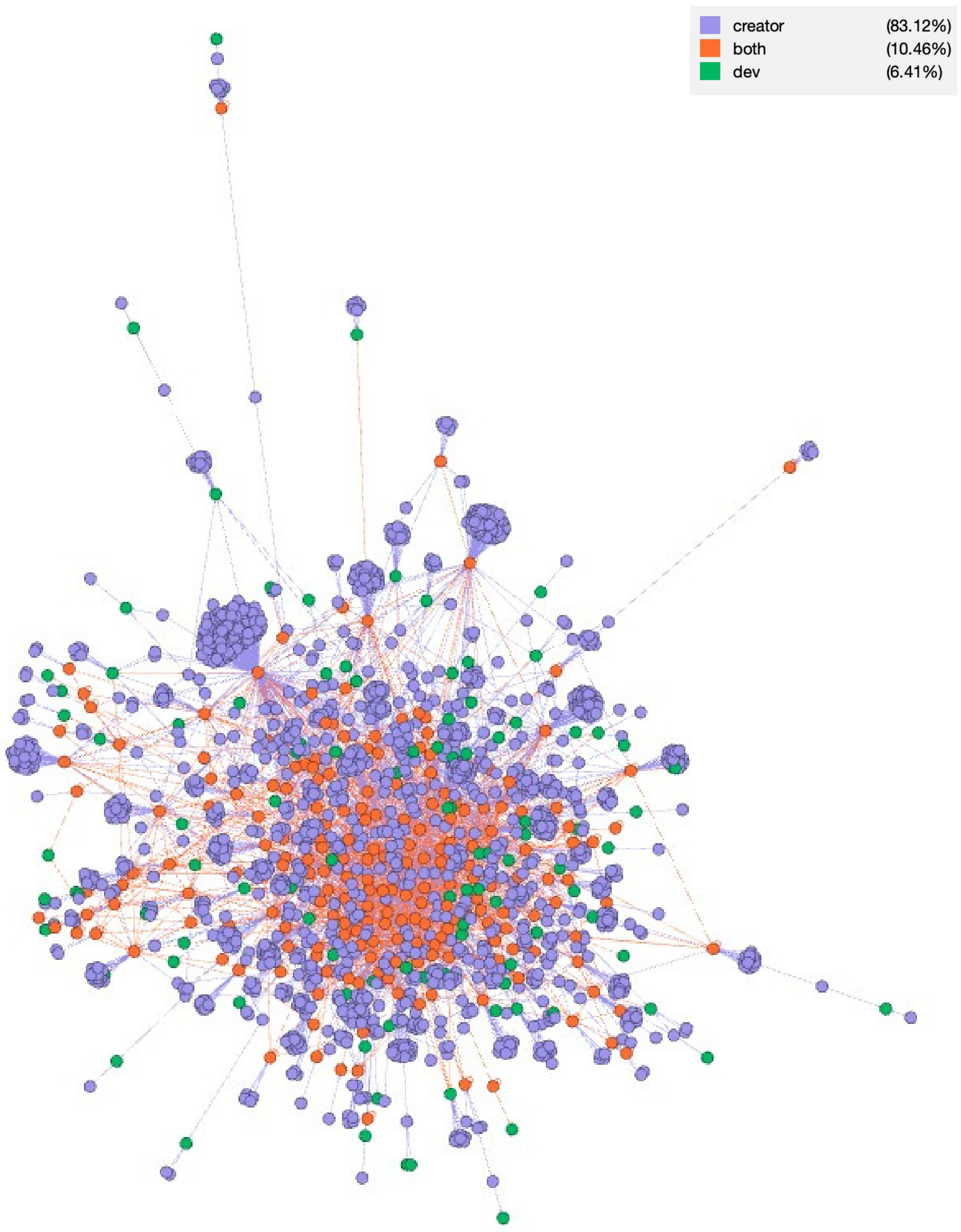


Figure 11

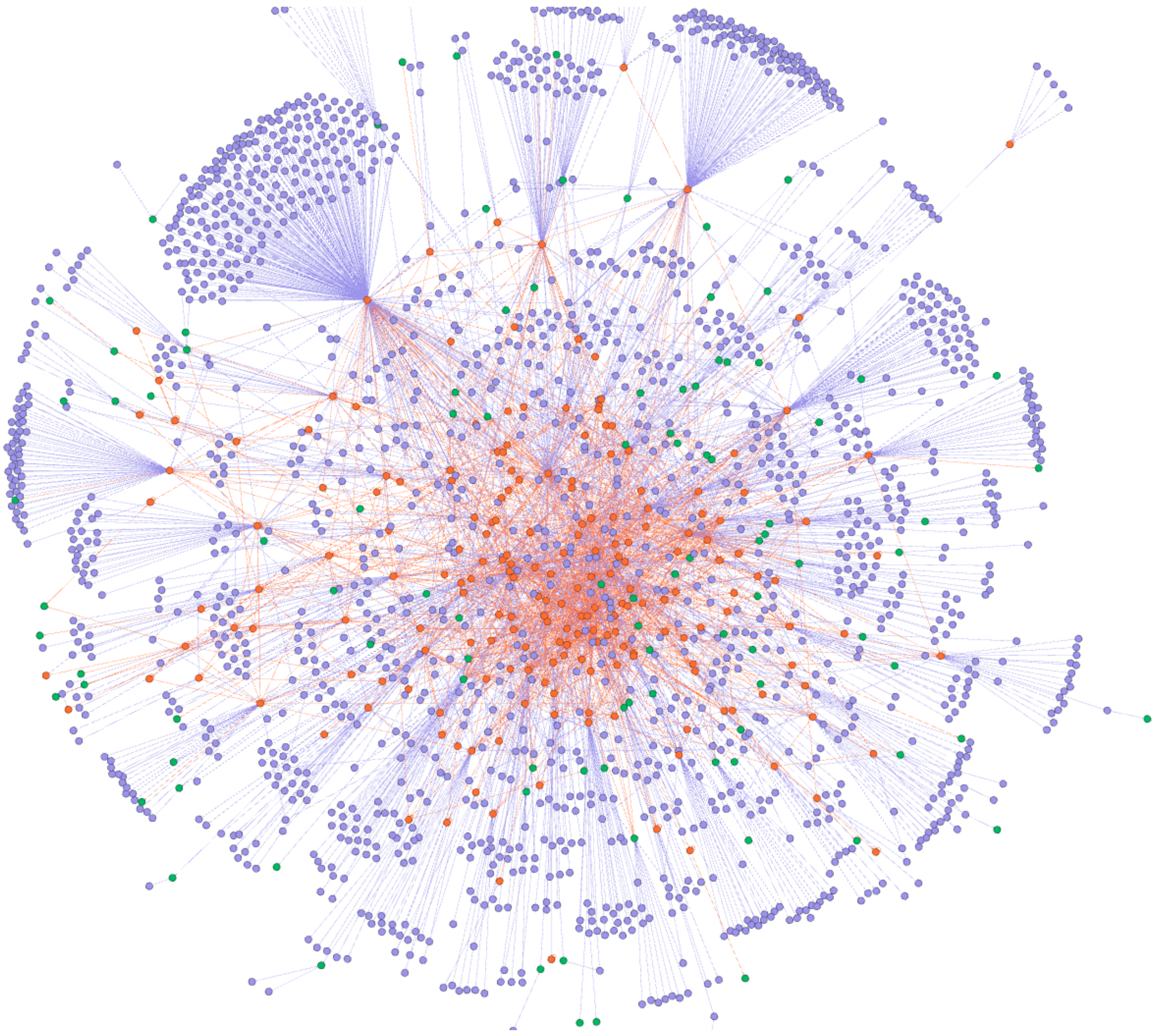
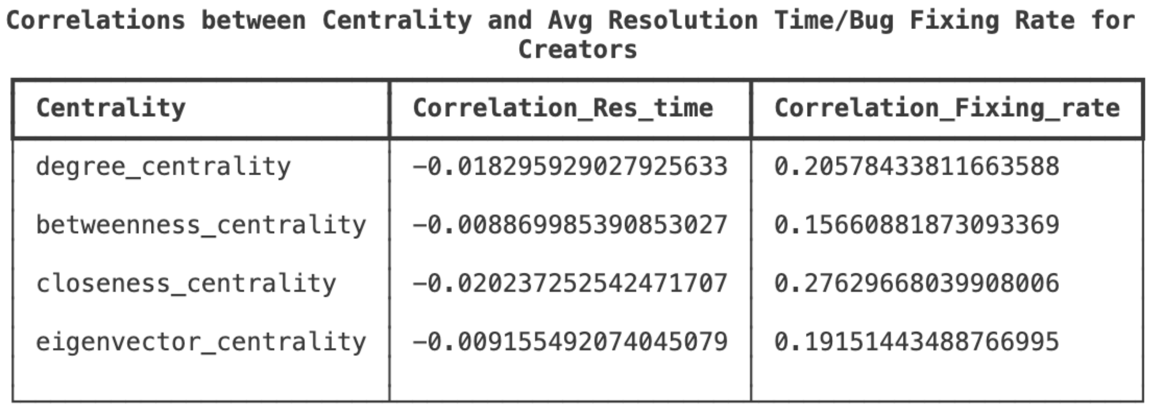


Figure 12

The visualization of our network graph reveals insightful characteristics about the Mozilla project's bug management process (blocker bugs). We can notice a typical structure found in many social and organizational networks, characterized by a densely connected core and a more sparsely connected periphery. Nodes in purple colour denote creators i.e. those who report blocker bugs but are not typically engaged in the resolution. Nodes in green denote developers, and nodes in orange represent individuals assuming dual role - contributing as both creators and developers. Firstly, the orange nodes, who function as both blocker bug reporters and resolvers, predominantly occupy central positions in the network. Their high centrality and high connectivity underscore their pivotal role within the network ecosystem. These individuals likely assume key leadership or senior roles, managing the lifecycle of blocker bugs from identification to resolution. Secondly, we can spot a few dispersed green nodes, indicating a small number of core developers associated with blocker bugs. Thirdly, the presence of distinct subgroups of purple nodes indicate the existence of specialized teams or units. These clusters, likely centered around specific products/features, reflect how certain key orange nodes coordinate the blocker bug reporting and resolution process within their specific operational areas, by acting as bridge between the core and peripheral nodes. However, the concentration of red nodes in central positions and the presence of peripheral connections for the purple nodes could lead to potential concerns regarding the project's diversity and inclusivity. Overall, the network graph provides a nuanced understanding of the organizational structure and operational workflow of blocker bug reporting and resolution within the Mozilla Bugzilla system.

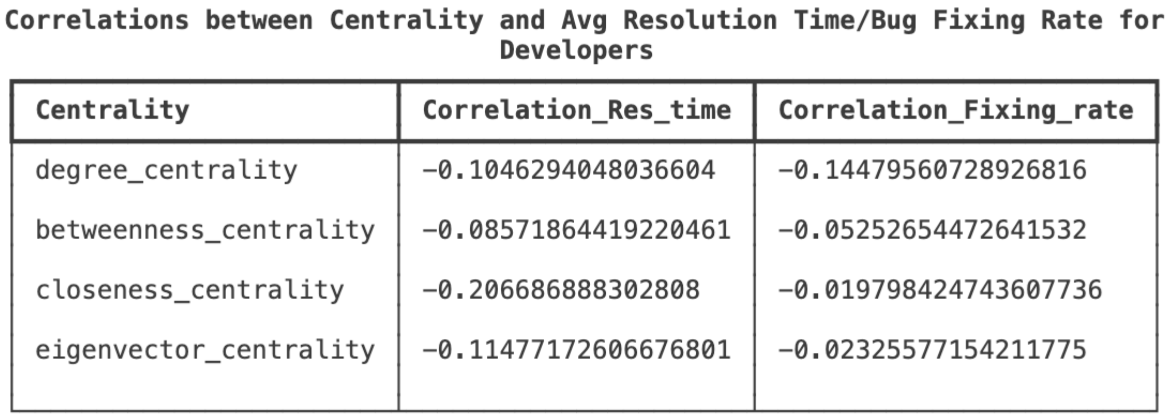
RQ5: Is there a correlation between the various centrality measures and bug fixing rate/average resolution time of creators and developers?



Correlations between Centrality and Avg Resolution Time/Bug Fixing Rate for Creators

Centrality	Correlation_Res_time	Correlation_Fixing_rate
degree_centrality	−0.018295929027925633	0.20578433811663588
betweenness_centrality	−0.008869985390853027	0.15660881873093369
closeness_centrality	−0.020237252542471707	0.27629668039908006
eigenvector_centrality	−0.009155492074045079	0.19151443488766995

Table 8



Correlations between Centrality and Avg Resolution Time/Bug Fixing Rate for Developers

Centrality	Correlation_Res_time	Correlation_Fixing_rate
degree_centrality	−0.1046294048036604	−0.14479560728926816
betweenness_centrality	−0.08571864419220461	−0.05252654472641532
closeness_centrality	−0.206686888302808	−0.019798424743607736
eigenvector_centrality	−0.11477172606676801	−0.02325577154211775

Table 9

When considering the correlation between degree centrality and the bug fixing rate for creators, a weak positive correlation of 0.205 is found. This suggests that creators with a higher number of connections (higher degree centrality) within the network demonstrate a marginally higher bug fixing rate. A similar trend is observed in the relationship between eigenvector centrality and the bug fixing rate for creators, where a correlation of 0.19 is found. Eigenvector centrality is a measure of a node's influence based on the number and quality of its connections. Hence, this suggests that creators connected to well-connected nodes may have a slightly higher fixing rate. Also, the relationship between betweenness centrality and the bug fixing rate for creators reveals a correlation of 0.16, indicating a very weak positive relationship. Betweenness centrality measures the extent to which a node lies on paths between other nodes. Therefore, creators who are more central to the network's flow might not necessarily have significantly higher fixing rates. The strongest relationship is seen between closeness centrality and the fixing rate for creators, with a correlation of 0.28. This moderate positive correlation suggests that creators who are 'closer' to others in the network, i.e., have a lower average shortest path length, tend to have a higher bug fixing rate. However, when examining the correlation between these centrality measures and the average resolution time for creators, the correlations are all very close to zero. This indicates almost no relationship between these variables.

When these centrality measures are correlated with the average resolution time for developers, all correlations are negative but weak, ranging from -0.08 to -0.21. Interestingly, closeness centrality demonstrates the strongest negative correlation of -0.21, implying that developers who are 'closer' to others in the network tend to have slightly shorter average resolution times. While examining the correlation between the centrality measures and the bug fixing rate for developers, the correlations are all very close to zero. This indicates almost no relationship between these variables.

These above findings suggest that although certain centrality measures are correlated with the bug fixing rate and average resolution time, the strength of these relationships is not very strong. While closeness centrality is somewhat significant, there might be other factors that play a substantial role in determining a creator's/developer's bug fixing rate and average resolution time.

Appendix:

- We chose *assigned_to* as the final developer, since the bug might be tossed around
- All the box plots are on log scale to account for high variation
- In Chapter 3 - part A, only top 100 creators have been considered during calculations
- While reading the report, please keep in mind that correlation does not imply causation. There could be other underlying factors or variables at play.

References:

- ur Rehman, M.H., Liew, C.S., Abbas, A., Jayaraman, P.P., Wah, T.Y. and Khan, S.U. (2016). Big Data Reduction Methods: A Survey. Data Science and Engineering, [online] 1(4), pp.265–284. doi:https://doi.org/10.1007/s41019-016-0022-0.
- Amrit, C., & Van Hillegersberg, J. (2010). Exploring the impact of soclo-technlcal core-periphery structures in open source software development. Journal of Information Technology, 25(2), 216-229.
- Baldwin, C. Y., & Clark, K. B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model?. Management Science, 52(7), 1116-1127.
- Zou, W., Xia, X., Zhang, W., Chen, Z. and Lo, D. An Empirical Study of Bug Fixing Rate. [online] Available at: http://www.mysmu.edu/faculty/davidlo/papers/compsac15-fixrate.pdf
- Dahlander, L., & Wallin, M. W. (2006). A man on the inside: Unlocking communities as complementary assets. Research Policy, 35(8), 1243-1259.
- Eiroa-Lledo, E., Ali, R.H., Pinto, G., Anderson, J. and Linstead, E. (2023). Large-Scale Identification and Analysis of Factors Impacting Simple Bug Resolution Times in Open Source Software Repositories. Applied Sciences, [online] 13(5), p.3150. doi:https://doi.org/10.3390/app13053150.
- Gulati, R., Puranam, P., & Tushman, M. (2012). Meta-organization design: Rethinking design in interorganizational and community contexts. Strategic Management Journal, 33(6), 571-586.
- He, V. F., Puranam, P., Shrestha, Y. R., & von Krogh, G. (2020). Resolving governance disputes in communities: A study of software license decisions. Strategic Management Journal, 41(10), 1837-1868.
- O’Mahony, S., & Ferraro, F. (2007). The emergence of governance in an open source community. Academy of Management Journal, 50(5), 1079-1106.
- O’Mahony, S., & Bechky, B. A. (2008). Boundary organizations: Enabling collaboration among unexpected allies. Administrative Science Quarterly, 53(3), 422-459.
- Von Krogh, G., Haefliger, S., Spaeth, S., & Wallin, M. W. (2012). Carrots and rainbows: Motivation and social practice in open source software development. MIS Quarterly, 649-676.
- www-archive.mozilla.org. Mozilla 1.1 Alpha Release Notes. [online] Available at: https://www-archive.mozilla.org/releases/mozilla1.1a/

- www-archive.mozilla.org. Mozilla 1.2 Beta Release Notes. [online] Available at: https://www-archive.mozilla.org/releases/mozilla1.2b/
- www-archive.mozilla.org. Mozilla 1.2 Release Notes. [online] Available at: https://www-archive.mozilla.org/releases/mozilla1.2/
- www-archive.mozilla.org. Mozilla 1.0. [online] Available at: https://www-archive.mozilla.org/releases/mozilla1.0
- Tian, Y., Lo, D. and Sun, C. (2013). Predicting Priority of Reported Bugs by Multi-factor Analysis. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICSM.2013.31.
- Uddin, Jamal & Ghazali, Rozaida & Mat Deris, Mustafa & Naseem, Rashid & Shah, Habib. (2017). A survey on bug prioritization. Artificial Intelligence Review. 47. 10.1007/s10462-016-9478-6.
- Saha, R.K., Khurshid, S. and Perry, D.E. (2014). An empirical study of long lived bugs. [online] IEEE Xplore. doi:https://doi.org/10.1109/CSMR-WCRE.2014.6747164.
- Nguyen, Tung & Nguyen, Anh & Tien, Nguyen. (2014). Topic-based, time-aware bug assignment. ACM SIGSOFT Software Engineering Notes. 39. 1-4. 10.1145/2557833.2560585.
- Jahan, S. and Rahman, M. Towards Understanding the Impacts of Textual Dissimilarity on Duplicate Bug Report Detection. [online] Available at: https://arxiv.org/pdf/2212.09976.pdf
- Yuan Tian, Nasir Ali, David Lo & Ahmed Hassan (2015). On the unreliability of bug severity data. core.ac.uk. [online] Available at: https://core.ac.uk/reader/35455730