International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 9 (2019) pp. 2260-2269

A Case Study on R: a powerful OSS and data analysis platform

Ranjan Kumar[1], Subhash Kumar[2], Sanjay K. Tiwary[3]

Abstract

R, an open source software, has emerged as a powerful tool for statistical computing and data visualization. It is also the favoured programming language for data science in the last decade. Its open source platform has allowed a tremendous growth in its community which has contributed R packages independently. Number of R packages has grown exponentially in the last five years. This paper undertakes a case study of R wherein the popularity of R and its development paradigm is explored, while studying salient features of its organization structure. Statistical inferences are drawn with respect to organization structure by performing statistical analysis of the available data on commits in R. The bug management system - a very important attribute of software quality is probed further while carrying out statistical analysis of the bug data of R. This supports the rigorous protocol of bug management system for OSS in general and R in particular.

Keywords: Commits, Bug management system, Data science, Software quality, Visualisation

1. INTRODUCTION

In recent times effective modeling, statistics, predictive analysis, data visualizations and powerful analytics is being harnessed by data scientists and leaders in making decisions and policies in a multitude of fields ranging from broad societal mapping, online marketing trends to the development of financial and climate change models which nurtures our economies and society. There is a very good chance that the R language is behind this number crunching exercise which is enabling transformation of data into useful information. R is open source software (OSS), which is a powerful tool and free statistical environment and programming language. R was initially developed by Robert Gentleman and Ross Ihaka, from the University of Auckland and is a descendent from the original "S" which was an early language for statistical analysis. Ihaka and Gentleman began their work on R software in the early 1990s as their personal project to provide a statistical environment for teaching of statistics. In the mid-1990s, they began to encourage others developers to join their project, which was eventually formalized in 1997 as the R Core Team. They adopted a strategy of combining the syntax of S, familiar to them and then in wide use among statisticians with the semantics of Scheme - a dialect of Lisp [1]. With the help of Martin Machler (working at the ETH Zurich), Ihaka and Gentleman released this software as free open-source software in 1995 [2]. Presently, the "Comprehensive R Archive Network" (CRAN) is the core of an increasingly growing R community [3] and it serves as the central repository for all resource materials related to R, including packages of functions, tutorials and discussion forums. The official public structure for the R Community is under the aegis of the R Foundation - a not for profit organization with a prestigious list of members and supporters. R is registered under GNU (General Public License). R is a programming language used by statisticians, data scientist and anybody who need to perform statistical analysis of data and process useful information from data using various mechanisms, such as classification, regression and text analysis. Currently, R has 782K lines of code which has increased from 82K lines of codes in 1997(about 35% C, 31% FORTRAN and 20% R) [4]. Changes in the code to base R are an exclusive domain of the core team members of R. Till now a total of twenty five members of R-core have contributed to R, Currently R-core has twenty members.

1.1 Characteristics of R

R is a programming language wherein data analysis is carried out by writing functions and scripts. However R is an easy language to learn. R is characterized by an extensive set of library functions which can implement not only mathematical but techniques of other disciplines like graphical, classification, clustering and statistics etc. R is easily extensible through functions, thus making R a robust interactive platform. It is a Vector based language, which permits application of functions to a whole vector in one operation, rather than having the requirement of an explicit loop. R can be considered as an interpreted language which can support large sets of data structures, which can be accessed directly through command line interpreter. R interprets the input code directly, without the need of a compiler by converting it into lower-level calls to pre-compiled code/functions. R supports a myriad of programming viz. generic, object oriented and procedural. Moreover, it supports seamless import of data from other applications. The heart of R is in R-packages which are libraries of functions. It empowers users to expand R's capabilities by introducing new packages which permits exploration of diverse fields including engineering and medical sciences. Till the last count, R had more than 12000 packages in its repositories providing a wide variety of statistical, machine learning and graphical

[1] *Department of Computer Science, Aryabhatta College,(University of Delhi), Benito Juarez Marg, New Delhi- 110021.; Email: ranjan301@gmail.com*
[2] *Department of Physics, Acharya Narendra Dev College,(University of Delhi), Govindpuri, Kalkaji, New Delhi- 110019; Email: subhashkumar@andc.du.ac.in*
[3] *Post-Graduate Department of Mathematics, Magadh University, Bodhgaya, Gaya-824234, Bihar.*

techniques. The machine learning techniques includes linear and nonlinear modeling, time-series analysis, classification, clustering and classic statistical tests. R embodies the entire gamut of data visualization tasks inclusive of data extraction, data cleaning, data loading, data transformation while performing statistical analysis and predictive modeling for data visualization.

R is said to be Turing complete, which implies it is computationally universal or complete in itself without the requirement of anything else in so far as data manipulation is concerned. R is equipped to work with indeterminate characters like undefined, missing and infinity values which do generally causes glitches in other traditional statistical platform. Documentation is the cornerstone of any good package, without which its versatility cannot be explored by the user. R has its own standard documentation format primarily based on Latex and rendered as HTML, plain text and pdf for the user. With a humongous increase in its list of packages, R has the capability to connect with other data stores, such as MySQL, SQLite, Hadoop and MongoDB for data storage activities.

The present paper undertakes a case study of R in its unique journey wherein with a modest beginning as an academic tool for teaching of statistics, R has now acquired the status of the most popular open source statistical analysis packages available in the market today. It commands support with a large and ever-growing user community who are adding new packages every day. The study explores its ever increasing popularity (Section 2) and the salient features of R development paradigm (Section 3). In Section 4, its rigorous bug management protocol is deciphered, while the last section (Section 5) presents the concluding remarks and scope for future investigation.

2. POPULARITY OF R

Since its inception, the development of R has been continuous and dynamic, thus metamorphosing into a powerful tool for statistical analysis and a true programming language. It enables competent users to add additional functionality by introducing a series of packages or by writing code to implement a new procedure. Moreover, all of the functionality embodied within R, or newly introduced are subject to continuous critique and improvement with respect to their accuracy, reliability and consistency. R core team maintains daily logs of code changes which makes it easier for users to track changes made in the past, present and future versions of R. One can also gauge the popularity of R by its appearance in scholarly articles/publications as an analysis tool or even as an object of study.

In 2018, R stood at 7th position amongst the most popular programming languages, whilst jumping from 9th position in 2014 in a ranking created by IEEE spectrum based on 11 metrics and 9 sources. It can be safely concluded that R is maintaining its momentum.

R occupied the top 10th position in in TIOBE index ranking 2018, thereby moving from the 24th position in the year 2013. The TIOBE index ranking is based on the number of hits (search) of turing complete programming languages on 25 most popular search engines such as Google, Yahoo, Bing, Wikipedia, etc [5]. Another barometer for popularity of a programming language is determined by how often tutorial of languages are being searched on search engines like Google. The PYPL popularity of programming language index does this and has placed R at the top 7th position [6]. In the arena of machine learning and data science, R has outplaced SAS as the most preferred tool as reported by the Burtch Work survey 2017. This survey found that among data scientists and predictive analytics professionals, R usage is 40%, while that of SAS is 34% [7].

R allows different users/developers to write packages and integrated functionality of R for specific tasks independently, without any intervention of R core members while making no changes in the R base software. This package system has not only helped R in its organic growth but also attracted a diverse group of users like those from medical science to engineering.

The distribution of R by CRAN began in 1997 with the advent of R core team. CRAN also contributed R packages on the internet. Today, after almost two decades, there are about 12000 packages on CRAN. Figure 1 depicts the growth in number of R packages on CRAN in the last ten years. It can be easily seen that the growth in the number of packages on CRAN in this period mimics the exponential curve, as can be seen from the curve drawn, by fitting the given data. This means that the growth increases at a consistent rate [8]. The value of R^2 (coefficient of determination) computed for the given data for the growth in number of packages on CRAN is found to be 0.9835. This is very close to 1, which further suggests that the exponential model is a good fit on the given R- package growth data.

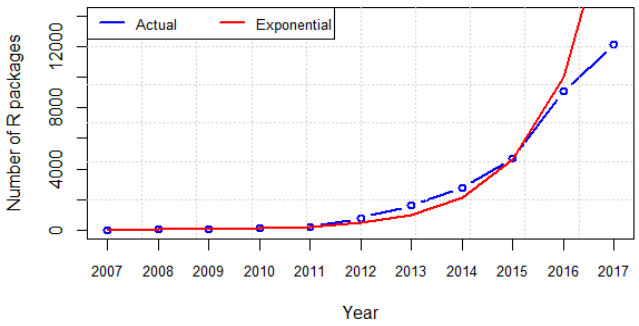

Fig. 1: Growth of the R package on CRAN

3. R DEVELOPMENT

R has achieved a phenomenal popularity in the last two decades as a true programming language and a powerful tool for data analysis in the rapidly growing field of data analytics. This achievement makes it an interesting subject of case study to investigate certain cardinal issues in its evolution. This is addressed, in this section by the investigation of following queries:

a) What is the development and organizational model of R?
b) How do the R developers coordinate among themselves when they are located in different parts of the world?

c) Do all developers contribute to the R software evenly?
d) Most of the core developers of R are found to be volunteers. Whether their contributions are made only during weekends or on holidays?

Who are those people who contributed to open source projects such as R? What motivates them to contribute to it? Is it completely philanthropic, or are there any latent rewards involved in open source development? Raymond [9] attempts to answers these questions by suggesting that there is indeed a "gift culture" in the open source development paradigm, wherein reputation is the chief currency. He further argues for the economic rationality of businesses enterprises which support open-source ecosystem. While in general, the R-core developers collaborate in the development and maintenance process of R, certain features of R development are handled by R-core members with specific interest and expertise in focused area. At the core, standard development and testing methodology are followed by R core members. This is followed by release of the source code to the R user community which is estimated to be around hundreds of thousands in number. This enables a comprehensive review, thus making R software more accurate, consistent and reliable.

R subversion is a version control based repository for the management of all source code of R with write access to source code granted to all R core members. To avoid any conflict in source code, release branch and ongoing development are segregated. To keep the process of review by the user community dynamic, daily logs of changes in codes (Commit) are catalogued and made available in public domain as R_svnlog_YYYY file at R developer website. In order to ensure quality of the highest order in the source code a well-defined process is adopted by the members while affecting changes in the source code and various testing mechanism. Utilizing known data and known results, a set of validation tests are undertaken and updated by R core members and errors if any, are resolved during testing before the release. In the Alpha or Beta release, the source code is available to user community for further/additional testing. Community communicate with core members through mailing list or via R bug tracking system.

3.1 Role and responsibilities of Core team

John Fox [10] has studied the evolution of R and has divided the life time of the R organization up till now into three stages. In first stage (1992-1994), Ihaka and Gentleman contributed to R development. In the second stage (1994-1997), many developers were recruited on invitation based on their credentials in the relevant area to actively participate and contribute to the development of R. Many of such recruited contributed voluntarily to R development. This can be considered as the transitional stage in the life history of R, which culminated with the formal setting up of the R core team with nine members in 1997. The period after 1997 till current date can be considered as a stage when the R core has been formally constituted and took over the charge of overall development to R. In February 2000, R 1.0.0 was officially released. Currently the R core team comprises of 20 highly qualified members, all are doctorates, except one, with non-profit motive. It is interesting to note that one among this group of twenty members in the R core team is an Indian.

The division of labor among R core members can at best be considered as fuzzy divisions, without any inherent rigidity However, some division of responsibility is assigned for critical objectives like maintenance and updating the versioning control, release management and maintaining CRAN package archive. For each of these tasks some members are made responsible formally. Since, the R-core members belong to ten different countries, effective communication and coordination across different geographical locations among R –core members is achieved electronically via e-mail. For this purpose a non-public email list is provided to them which also serve as a discussion forum among the core members. The R core members do meet regularly at their schedule conferences specific to Statistical computing and R which provide opportunity to discuss the progress of ongoing project [11]. This also enables proper and effective coordination of the ongoing project.

Since write access to the source code of R is restricted to only the R-core members, it is interesting to know here whether each R-core member contributes evenly in so far as changes in source code (known as commit). For this investigation, we have accessed the daily log subversion file, R_SVNLog from the developer website [12]. The log file contains the commit data for the period of 1.1.2013 to 12.02.2017. Here we extracted commits made by each core members separately as shown in Table 1.

Table 1: Number of commits made by each R core members (committer)

Year → / **R-core member ↓**	**2013**	**2014**	**2015**	**2016**	**2017**
Ripley	2134	1676	1297	671	239
Kalibera	0	0	0	11	34
Maechler	169	281	434	503	71
Gmbecker	0	0	0	0	8
Hornik	186	232	343	212	14
Jmc	21	8	7	33	10
Murdoch	318	177	147	125	5
Lawrence	0	22	66	90	5
Pd	70	70	98	114	13
Luke	121	155	79	199	14
Morgan	0	9	28	28	2
Ligges	19	3	2	18	0
Urbaneks	26	25	5	8	0
Murrell	15	12	27	9	3
Plummer	0	10	7	9	0
Deepayan	14	4	2	0	0
Duncan	0	1	0	0	0
Leisch	13	0	0	0	0
Jangorecki	0	0	0	3	0

The information contained in Table 1 is also depicted in the figure 2 where number of commits made by each core members is shown as a function of time (calendar year). It is easy to conclude from this figure that only few of the members are very active as far as committing is concerned. Gini coefficient, a statistical measurement of inequality, was calculated on the number of commits made. Its value ranges from 0(0%) to 1(100%) where 0 indicates perfect equality and 1 for perfect inequality. The year wise Gini values are illustrated in the figure 3. It can be seen that the Gini coefficient is in the range of 0.8 to 0.9, which indicates the distribution of commits made by R-core members is highly unequal.

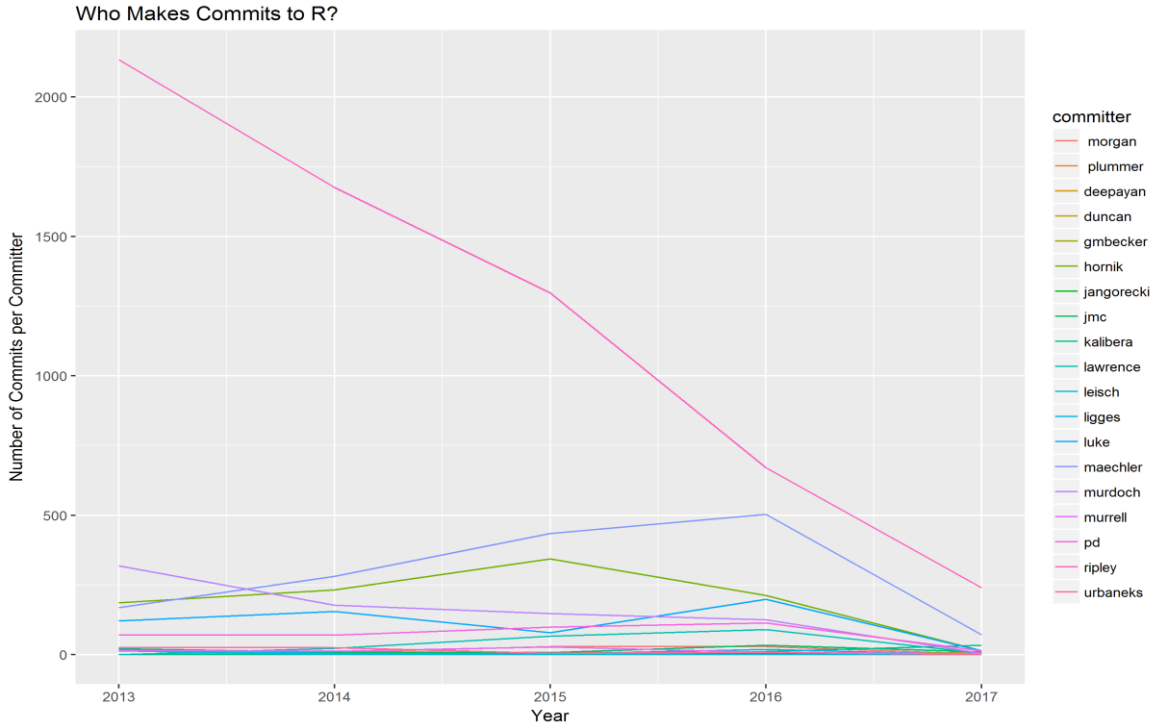

Fig. 2: Number of commits made by each committer (R core member) from the year 2013-2017

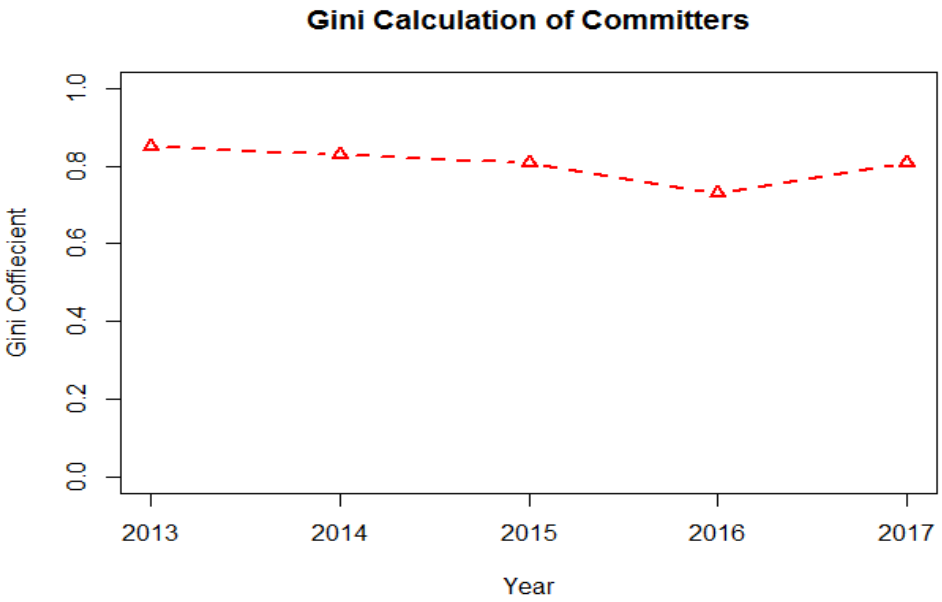

Fig. 3: Gini coefficient of the committers

Another aspect of working pattern of R developer is pattern of contribution to commits among the various days of a week. This is investigated with 10667 commits data. It was found that commits made by the R core developers were evenly distributed from Monday to Sunday which is also depicted in figure 4. If it were uneven, then this might affect the quality of the software. Maelik et.al. [13] has also observed, taking data from 79 large successful open source software, similar pattern in the working culture of all OSS developers.

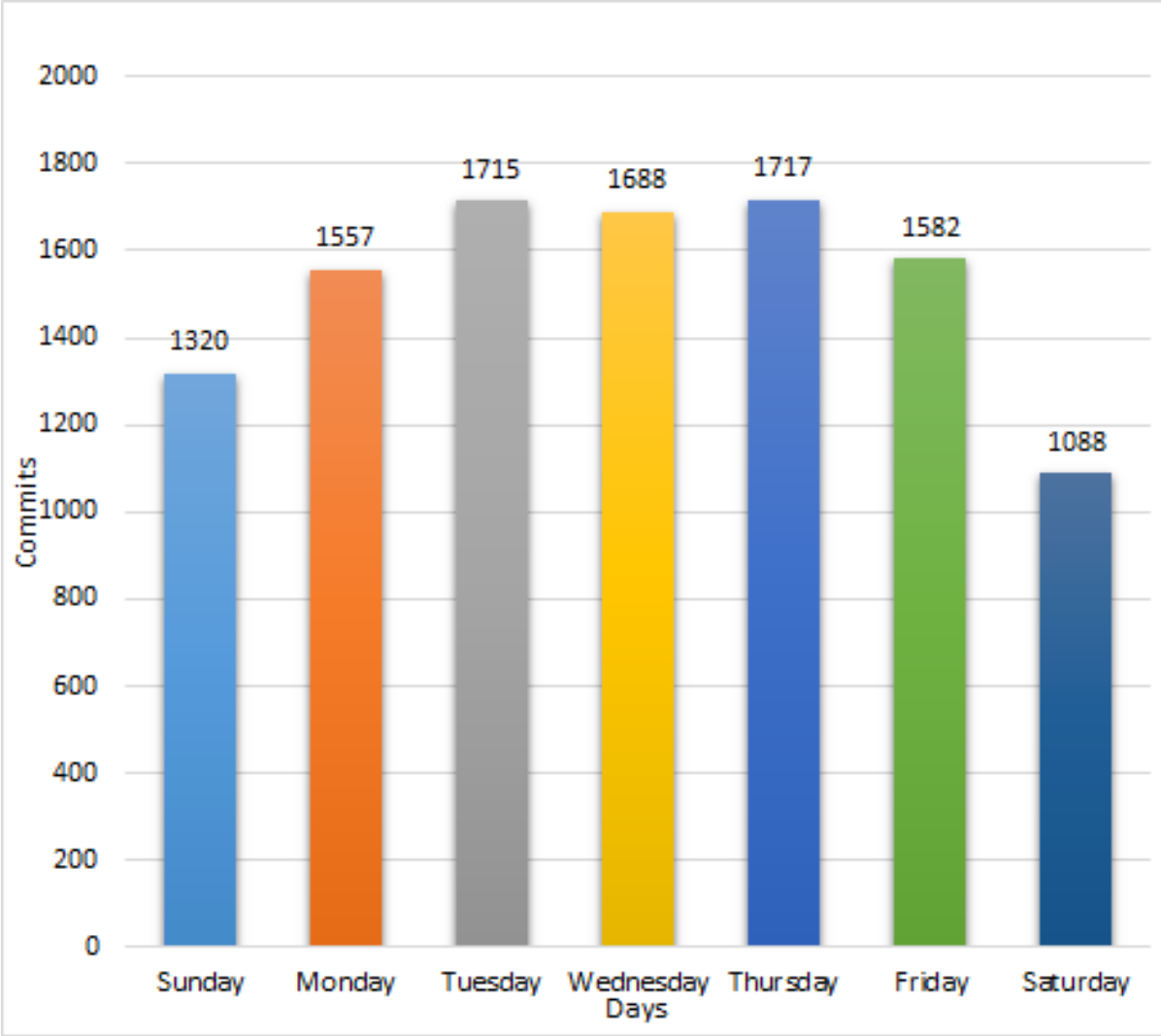

Fig. 4 Distribution of contribution made by the committers (R core members) in weekdays.

4. BUG MANAGEMENT IN R

Bug Life Cycle is a cycle of events which a defect of the software goes through during its entire lifetime. It starts when the defect is discovered and culminates when that defect is closed, while ensuring it's not reproduced further. Bug life cycle refers to the bug found during testing. Within the Life Cycle, the bug has different states. Bug Management is very important aspect in improving quality of software. Software maintenance requires a large amount of cost and efforts. More than 90 % of the cost and efforts is spent on maintenance activities [14]. A sizeable part of the resources of software maintenance are earmarked for Bug management. It includes bug detection, assignment of bug to developer for fixing and Bug correction. This section has dealt with the following research questions vis-à-vis Bug management in R:

(a) What is R bug life cycle?
(b) Whether and how much external users/developers involved in bug reporting and bug correcting?
(c) Among the bugs resolved, how many among them are valid bugs?
(d) How fast bugs are resolved as a whole and as component wise?
(e) Whether priority assigned to the bug is related to bug resolution time?

Figure 5 has shown schematically the majority of states of a bug in a typical bug life cycle. Bugs are reported by developers, testers or users in the bug racking system after the discovery and identification of any bug. If a first time user (not a R member) reports the bug it is assigned to unconfirmed state

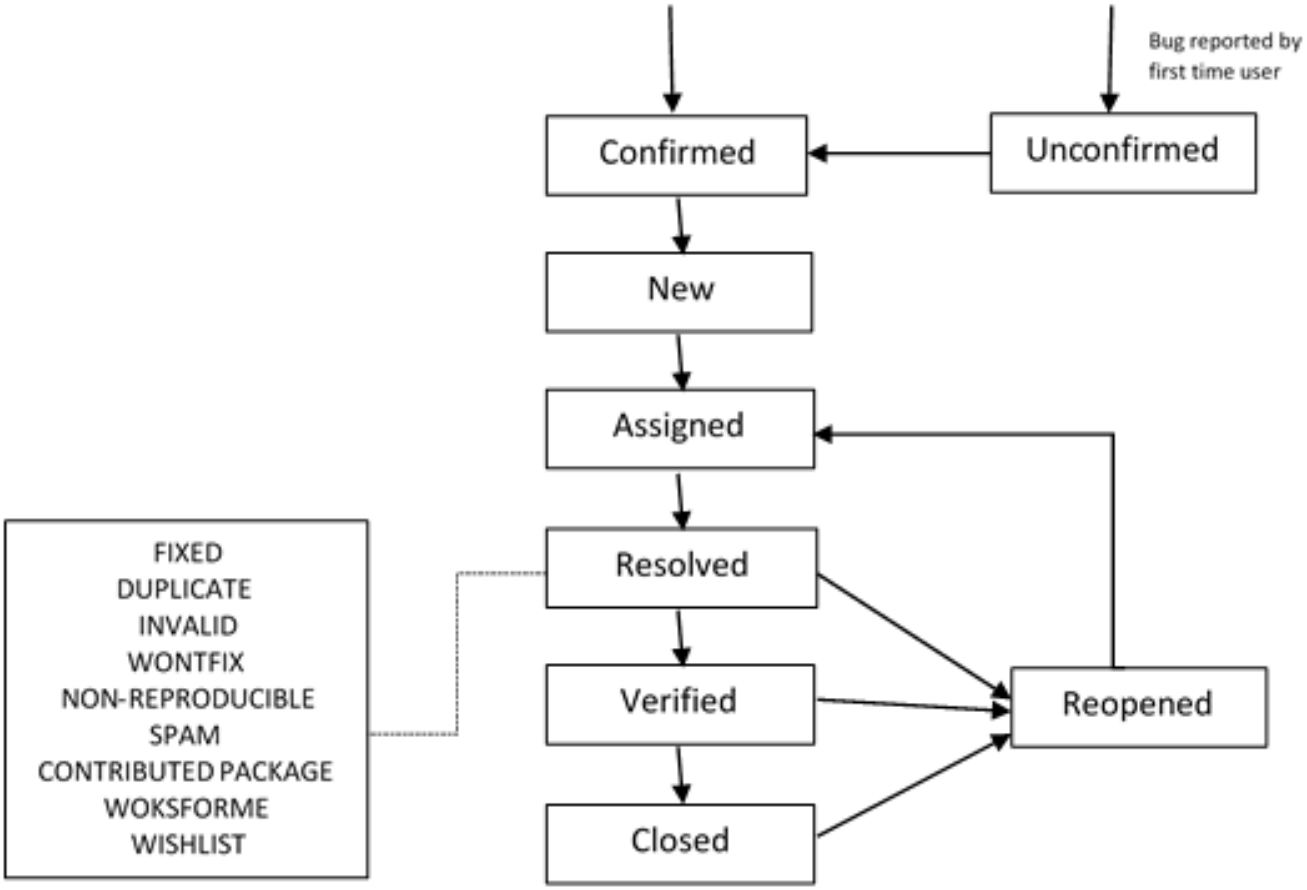

Fig. 5: Schematics of Bug Life Cycle in R

otherwise the reported bug moves to the confirmed state. Unconfirmed bug is reviewed and if found to be relevant, then it is updated to the confirmed state, or else its resolution is updated to the Resolved state. Next, the bug is triaged and its state is updated to *New*. It then gets assigned to most qualified developer who can fix it (i.e., it assumes the state *Assigned*). Developer then fixes the bug and updated into resolved FIXED state. Fixed bug are then verified further by another developer/tester and state updated to Verified state. After verification it moves to the closed state. Other resolutions are undertaken when it is not a software bug (INVALID), when the same bug is reported earlier (DUPLICATE), when the bug is valid and reproducible but will not be fixed because of lack of resources or low priority (WONTFIX), when the bug is not reproducible by the developer (NOT REPRODUCIBLE), if bug required some more information to fix later (WORSFORME), when bug reported due to defect in packages (CONTRIBUTED PACKAGE) or when a bug is basically a suggestion for future enhancement in the software (WISHLST).

4.1 Bug Data Quantification

To further explore the bug management, bug data quantification is carried out by utilizing data extracted from R's Bug tracking system, Bugzilla [15] - commonly used to manage bug resolution process. It provides data for bug opening date, bug resolution date, components, version, assignee, etc. A total of 6202 bugs were extracted on 04.04.2017 from Bugzilla bug tracking system where 5640 bugs are found to be in the close state. First we analyzed the data with two columns - reporter and assignee, to know the number of users other than R core members, who have acted as reporter and developers in the resolution of the bug. We found that while 1396 different users have reported the bugs to bug tracking system, only 44 unique developers, including 35 from outside have contributed in resolving the bugs.

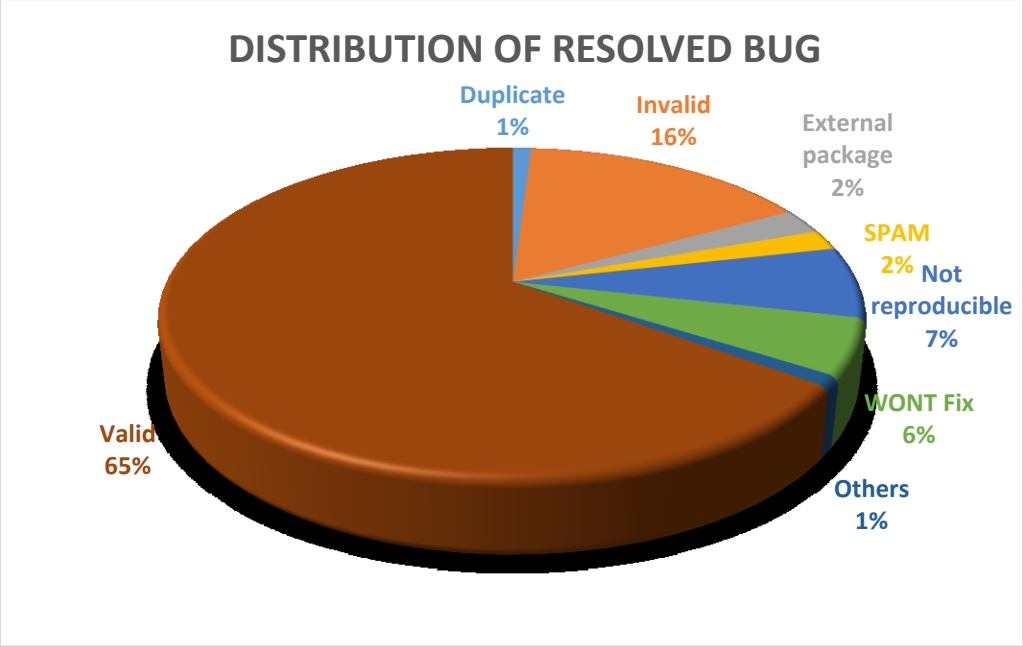

Fig. 6: Distribution of resolved bug which finally changed to close state.

Out of these 5640 resolved bugs, only 65% are valid bugs, 16% are invalid bugs, around 7% are not reproducible, 6% bugs won't be fixed, 2 % bugs are contributed from external packages, 1% duplicate bugs, 1 % are bugs comprises the others including wishlist, workforme and around 2% are SPAM. The distribution of the bugs reported in the close state are shown in the pie diagram in Figure 6.

Bugs are categorized into 18 components. We have analyzed these data after calculating bug resolution time (days) by considering calendar date. The calculated R bug resolution interval overall as well as component wise is approximated in days to percentage of bugs resolved within that interval. As shown in Figure 7, 61 % of the bugs are resolved on the same day, 66% in one day, 76 % bugs are resolved in 10 days and 90% within 1000 days. Compared with Apache – the leading web server, first 75% of the bugs are resolved much faster in R but the last 25 % bugs are fixed quickly in Apache as compared to R [16, 17].

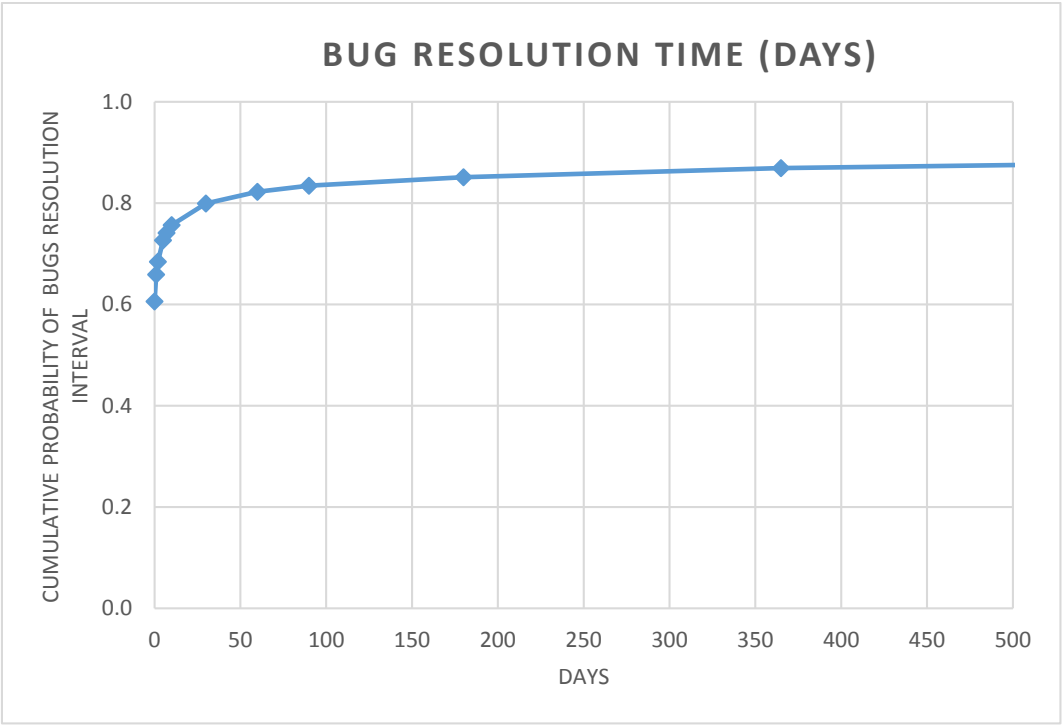

Fig. 7: Cumulative Bug Resolution over days

We further explore to find out whether these numbers are dependent on priority. From the Bugzilla database, it can be seen that more than 95 % of the bugs are accorded highest priority and P5 set (5550 of 5640) in the priority field. It is also found that amongst the longest period taken to resolve the bugs, more than top 30 are having highest priority level. This brings out the fact that priority defined in the database has no effect on bug fixing time. This may have perhaps happened because the priority is being set by the reporter at the time of reporting the bug. Mockus et al. also reported the same and argued that commercial software priority is strongly related to bug resolved time [17].

Bugs in bug tracking system are categorized into components which tend to reflect priorities since they reflect number of developers affected. Figure 8 shows comparisons among such groups of components. The horizontal axis shows the resolution time in days and the vertical axis shows the cumulative probabilities of bugs within that time interval.

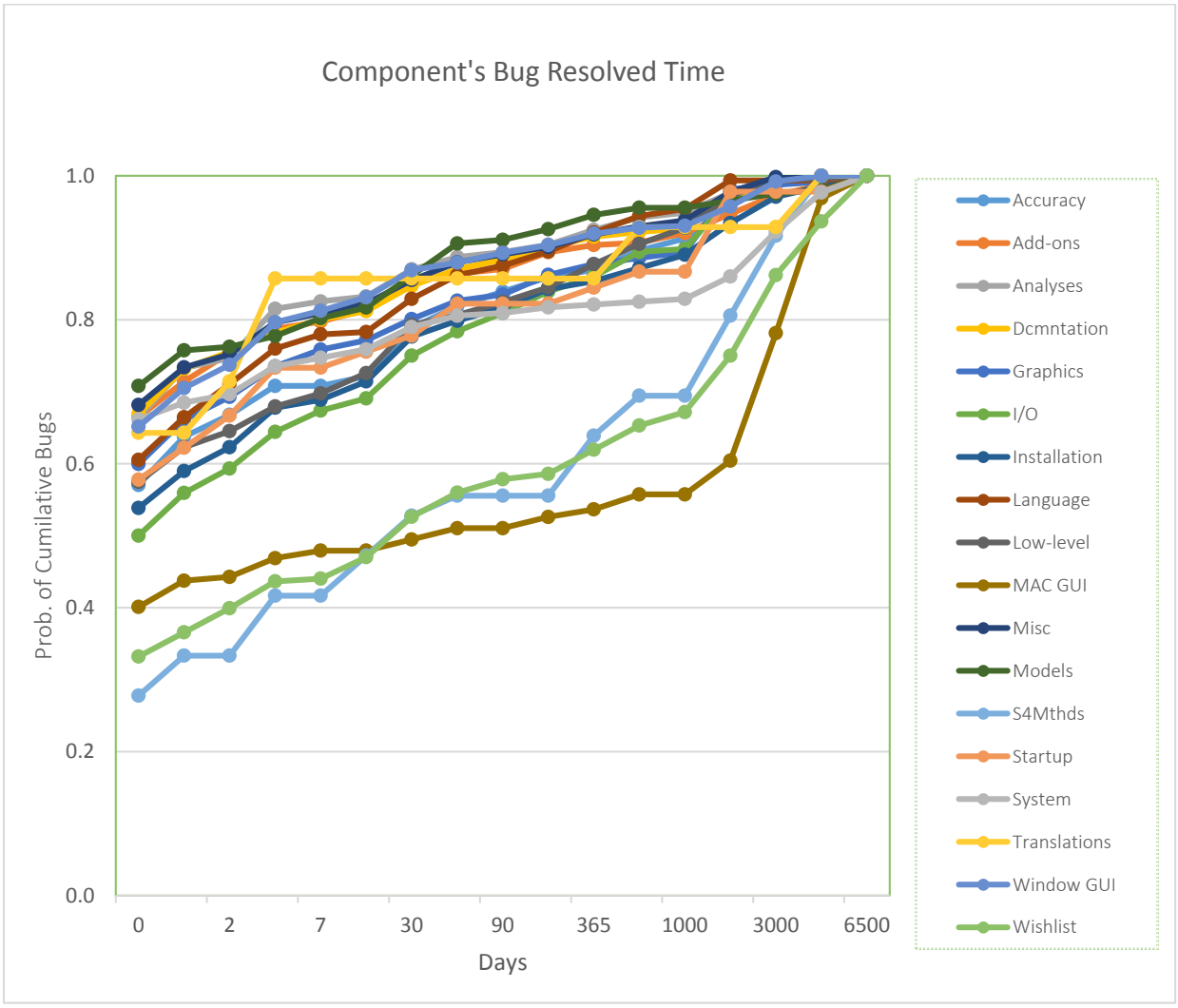

Fig. 8: Component wise bug resolved time

Bugs related to System, models, transactions, installation, etc. components is accorded higher priority because they involve many users and could potentially cause major failures. On the other hand Wishlist, S4 methods and MAC GUI components are not widely deployed so they do involve lesser number of users. From the user point of view wishlist, S4 methods and MAC GUI components should have been assigned lower priority. The present data (Figure 8) show exactly this pattern - having much faster closure times for the higher-priority problems. The differences between the trends in the two different groups are statistically significant as can be discerned by the calculated *p*-value of less than 0.05. This has been tested using Tukey Honest Significant Difference (HSD) method with 95 % of confidence level, whereas the trends within groups do not differ significantly. The boxplot shown in Figure 9 also follow the same pattern.

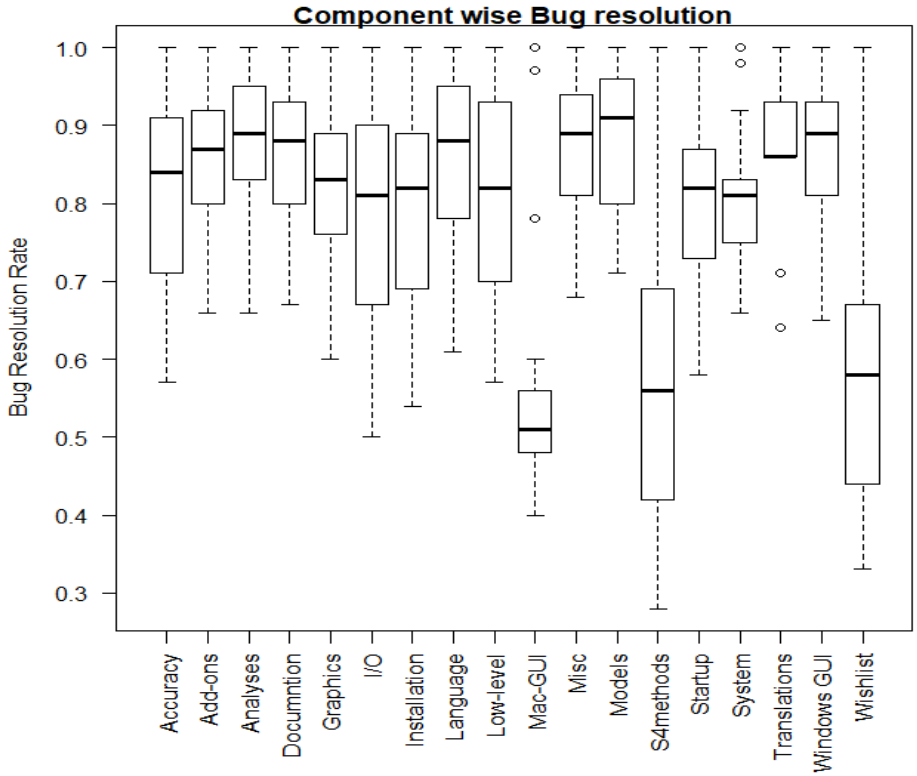

Fig. 9: Boxplot of component wise bug resolved data

4.2 Security and Recovery

R servers are physically situated at such institutional infrastructures which are endowed with well-defined security policy with respect to accessing and feeding the server. To provide logical security, core members are provided with user name and password to enable access to the computing systems. Yet it's not Carte blanche, the standard security protocol provide users limited access to the system depending upon functional requirements. Networking is also well secured using security policies and related mechanisms by employing software controls like firewall, etc.

5. CONCLUSION AND FUTURE SCOPE

In last decade R has not only emerged as a powerful statistical computing tool but also as a strong procedural language to be used in the field of Data science, Big data, medical applications etc. In this paper we have undertaken a case study of R, a powerful OSS, in order to calibrate its immense popularity vis-à-vis other platforms for statistical analysis by probing as well as exploring their organization structure, development paradigm of R, coordination among developers and the R bug management system. Statistical analysis is carried out to draw quantitative statistical inferences of R's development and the bug management system which supports its stature of being one of the most powerful statistical computing platform and a programming language for data science. There are certainly some aspects which could not be considered in this research paper and can be a subject of future investigation. There is a scope to study the details of the source code which were completely left out in the work presented here. Additionally, the availability of the bug data can also allow taking up reliability study in details including the reliability models.

REFERENCES

[1] Fox, John & Leanage, Allison, 2016. "R and the Journal of Statistical Software," Journal of Statistical Software, Foundation for Open Access Statistics, vol. 73(02)

[2] Ihaka, R., Gentleman, R., 1996. R: a language for data analysis and graphics. J. Comput. Graph. Stat. 5, 299–314.

[3] Hornik, K., Leisch, F., 2001. Vienna and R: Love, Marriage and the Future. In: R., Dutter (Ed.), Festschrift. 50 Jahre Osterreichische Statistische Gesellschaft. ¨ Osterreichische ¨ Statistische Gesellschaft, Vienna, Autriche, pp. 61–70.

[4] https://www.openhub.net/p/r_project/analyses /latest/languages_summary; retrieved on 2.1.2018.

[5] https://www.tiobe.com/tiobe-index/; retrieved on 10.7.2018.

[6] http://pypl.github.io/PYPL.html; retrieved on 10.7.2018.

[7] https://www.burtchworks.com/2017/06/19/2017-sas-r-python-flash-survey-results/; retrieved on 10.7.2018

[8] R Core Team, 2017. R: A Language and Environment for Statistical Computing, https://cran.r-project.org/

[9] E. Raymond, 2001,The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, pages 65–112. O'Reilly, Sebastopol CA

[10] Fox John, 2009, Aspects of the Social Organization and Trajectory of the R Project, The R Journal Vol. 1/2.

[11] https://www.r-project.org/conferences.html

[12] R Core Team,2017. R: A Language and Environment for Statistical Computing, http://developer.r-project.org/R_svnlog_2013. Retrieved on 13.02.2017

[13] Maëlick Claes, Mika V. Mäntylä, Miikka Kuutila, and Bram Adams. 2018. Do programmers work at night or during the weekend? In Proceedings of the 40th International Conference on Software Engineering (ICSE '18). ACM, New York, NY, USA, 705-715. DOI: https://doi.org/10.1145/3180155.3180193.

[14] Erlikh L,2001. Leveraging legacy systems in modern architectures. White paper from Relativity Technologies

[15] https://bugs.r-project.org/bugzilla3; retrieved on 04.04.2017

[16] Kumar, Ranjan, Subhash Kumar, Sukanto Dev, "Adoption and evolution of FOSS: key factors in the development of the Apache web server", in proceedings of Recent Advances in Mathematics, Statistics and Computer Science, World Scientific, pp. 362-370'

[17] Mockus, A., T. Fielding, and D. Herbsleb. Two case studies of open source software development: Apache and Mozilla.ACM Trans. Software Engineering and Methodology, 11(3):309–346, July 2002.