Received: 9 March 2020 | Revised: 13 April 2020 | Accepted: 22 April 2020

DOI: 10.1002/smr.2270

SPECIAL ISSUE - EMPIRICAL PAPER

Journal of Software: Evolution and Process WILEY

Evolution and prospects of the Comprehensive R Archive Network (CRAN) package ecosystem

Marçal Mora-Cantallops | Miguel-Ángel Sicilia | Elena García-Barriocanal | Salvador Sánchez-Alonso

University of Alcalá, Alcalá de Henares, Spain

Correspondence
Marçal Mora-Cantallops, Universidad de Alcalá, Alcalá de Henares, Spain.
Email: marcal.mora@uah.es

Abstract

Free and open source software package ecosystems have existed for a long time, but such collaborative development practice has surged in recent years. One of the oldest and most popular package ecosystems is Comprehensive R Archive Network (CRAN), the repository of packages of the statistical language R, a popular statistical computing environment. CRAN stores a large number of packages that are updated regularly and depend on many other packages in a complex graph of relations. As the repository grows, its sustainability could be threatened by that complexity or non-uniform evolution of some packages. This paper provides an empirical analysis of the evolution of the CRAN repository in the last 20 years, considering the laws of software evolution and the effect of CRAN's policies on such development. Results show how the progress of CRAN is consistent with the laws of continuous growth and change and how there seems to be a relevant increase in complexity in recent years. Significant challenges are raising related to the scale and scope of software package managers and the services they provide; understanding how they change over time and what might endanger their sustainability are key factors for their future improvement, maintenance, policies, and, eventually, sustainability of the ecosystem.

KEYWORDS

CRAN, empirical analysis, open-source software, package ecosystem, R, software evolution

1 | INTRODUCTION

In recent years, new computing technologies and the availability of large volumes of data have driven a surge in the creation and development of new methods to analyze data. Such methods are used across many disciplines, as they have become common tools for researchers and practitioners. The rise in computational power has not only facilitated the creation of newer and more complex methods but also allowed to gather large amounts of information, that, in turn, generated new methodological approaches, while easing the access to the use of such new models. The result of this feedback loop has been an increase in the usage of programming languages for statistical computing such as R, which, as of February 2020, ranks 13th in the TIOBE index (https://www.tiobe.com/tiobe-index/), a measure of popularity of general purpose programming languages.

The R programming language is widely used among statisticians and data miners for developing statistical and data analysis libraries, while also being one of the most popular languages among data scientists thanks to its flexibility and expansion capabilities, as R can be extended through user-created packages. The Comprehensive R Archive Network (CRAN) (https://cran.r-project.org/) is a network of web servers around the world where R source code, R manuals, documentation, and contributed packages can be found, and it can be considered as the official repository, containing the largest collection of available R packages. At the end of 2019, it hosted a total of 15,368 packages.

Software ecosystems such as CRAN could be considered as very complex networks of artifacts, product of the exponential increase in collaborative development under the open-source software (OSS) paradigm in the last two decades. Various researchers have compared software ecosystems with natural ecosystems[1] as they also they grow and evolve. The evolution of software, however, has been studied specifically, and there is a well-established set of empirical observations about how individual software systems evolve.[2] Lehman's laws have been used and verified in multiple studies over different contexts; for instance, in recent years, Gezici et al.[3] applied Lehman's framework to explore internal quality, external quality and the relation between these two in the evolution of mobile software. Braga de Vasconcelos et al.[4] also adopted Lehman's stance on software evolution, arguing that the adoption of knowledge management practices in software engineering would improve both software construction and more particularly software maintenance. Svahnberg and Gorschek,[5] in a similar manner, recognized that as software systems grow, it becomes more and more costly to develop them from scratch for every new system and that according to the laws of software evolution, as this happens, it becomes impractical to recreate software that may be readily available already. In this context, it is relevant to analyze software systems and ecosystems to maximize not only the reuse but also the value of doing so, selecting the best candidates to increase their potential, optimize maintenance efforts, and, in general, reduce the associated risks.

The structure of OSS projects, with CRAN not being an exception, is often referred to as being a 'hierarchical onion–like structure',[6] which consists of core developers, codevelopers, active users, and passive users. This kind of structures are not uniform in their distribution, and the core contributors typically account for a very small portion of the number of developers and users, and, also in contrast with typical closed source software development, many times contribute to OSS projects for free on top of their regular job or as a hobby. In this context, the lack of maintenance, uncontrolled growth, or increasing complexity may result, gradually, in reduced usefulness for its users, contributors leaving, knowledge destruction, and, eventually, abandonment of the OSS project.

Previous works also show how Lehman's laws are applicable[7] to analyze the ecosystem-level evolution of large repositories that are built by the collaborative efforts on a single common project and, therefore, could be helpful to identify where the maintenance efforts should focus, introduce new policies to avoid falling in decreasing perceived quality, and, in general, assess the health of the ecosystem.

Thus, and based on the studies described above, this work aims to empirically explore the evolution of the CRAN package ecosystem in regard to Lehman's laws as a representative OSS project. To do so, a number of metrics available in the literature will be collected and analyzed to assess its adherence to the laws of software evolution in the last two decades, since CRAN's inception to the current day, while also looking at the future prospects and risks for the ecosystem.

Overall, the current analysis demonstrates how the CRAN package ecosystem has followed a continuous change over its lifetime, with seven to eight times as many updates as new releases and with older packages still being regularly updated. There is a continuous growth both in absolute figures (number of packages) and in more relative terms (links between packages in the ecosystem), although part of this growth seems to be resulting in a notable increase in complexity over the most recent years that could represent a future risk for the health of the ecosystem as it keeps growing.

The remainder of this article is structured as follows. Section 2 discusses related work. Section 3 includes the research methodology together with the data extraction and analysis methods. Section 4 addresses the results of the analysis, while commenting on each law. Section 5 adds perspective to the results, linking them to the related work and discussing this work's contributions. Section 6 finally concludes, with an additional mention to future related lines of work.

2 | BACKGROUND

2.1 | The laws of software evolution

Belady and Lehman started studying software evolution in a systematic way in the late 1960s,[8] developing a set of empirical laws that grew from three initial laws to a total of eight at the end of the 1990s.[2] Barry et al.[5] summarized Lehman's work and ordered them into three broad categories: laws about the evolution of software system characteristics, laws related to organizational or economic constraintsm and meta-laws of self-regulation and feedback. The current article will be focused on the first set of laws, which look at the characteristics of the evolution of a software system and correspond to the Laws I, II, VI, and VII of the original enunciation by Lehman. These four laws are known as the laws of continuing change, increasing complexity, continuing growth, and declining quality, respectively.

- Continuous change: Systems must continually adapt to the environment to maintain satisfactory performance.
- Increasing complexity: Functional content of systems must be continually increased to maintain user satisfaction.
- Continuing growth: As systems evolve they become more complex unless work is specifically done to prevent this breakdown in structure.
- Declining quality: System quality declines unless it is actively maintained and adapted to environmental changes. This law is considered by Herraiz et al.[9] as a corollary of the previous ones, 'as the decline of quality is due to the degradation of the system under continuous change'.

Many works have looked at how Lehman's laws apply to various contexts and software; support has been strongest for the laws of continuous change, continuing growth and increasing complexity.[10] Herraiz et al.[7] conducted a systematic literature review summarizing both the evolution and the changes in the laws themselves over the years and the results of multiple studies. They found that the laws of continuous change and continuing growth were clearly supported by all the studies, while the law of increasing complexity had been inconsistent with evidence in many cases, as it was affected by the growth pattern of the projects; software projects whose growth accelerated over time were not coherent with that law. Both reviews, however, agree on considering the declining quality law the most difficult to prove empirically, although some results show a certain degree of support.[10,11]

2.2 | Laws of software evolution in OSS

The first studies attempting to verify the laws of software evolution for OSS were based on a relatively small set of case studies.[12] A few years later, a large-scale investigation on 8,621 OSS projects by Koch,[13] albeit limited by the characteristics of the sample, revealed that a majority of projects evolve according to the behavior predicted by the laws, but around 40% of them grew in a superlinear pattern, incompatible with the laws. He speculated that the cause of such incompatibility was a certain organizational model, the chief programmer team, where a limited number of developers concentrated most of the work. Israeli et al.[14] studied 810 versions of the Linux kernel, aiming to find out whether the laws were followed and concluded that those related to growth and stability of the process were supported. However, the main conclusion of their study for the current work's purposes is that the perpetual development model of OSS software is compatible with the laws of software evolution; thus, it seems appropriate to use them as the framework for the study of CRAN.

In 2008, Fernandez-Ramil et al.[15] published an article where the main empirical studies about the evolution of OSS were reviewed, and their findings summarized in regard to the confirmation (or not) of the cited laws. It is worth noting that the laws do not apply to OSS projects that remain in initial development or in proposal stage. The applicability of Lehman's Laws I, II, VI, and VII to successful OSS projects is reproduced in Table 1 (the other four laws are also available in the cited work).

The observations summarized by Fernandez-Ramil et al.[15] also highlight how continuous change and continuing growth can be tested, but increasing complexity and declining quality are controversial. In the first case, if an increasing complexity is observed, it fits the premise of the law, but if it is not observed, then it might be due to the work that has been done to reduce (or prevent) it, thus also satisfying the law. In the second, the opposite happens; if quality declines, it fits the law, but if it does not, it could be due to the maintenance and adaptation efforts). Thus, Laws II and VII can be observed but not discarded.

More recent works highlight the importance of research on the maintenance of OSS and potential risks that they experience. Coelho et al.[16] proposed a machine learning model to identify unmaintained GitHub projects and to measure the level of maintenance activity of active GitHub

TABLE 1 Applicability of the laws of software evolution to successful OSS projects

Number	Name	Comment on applicability to OSS evolution
I	Continuous change	Seems to apply well to those OSS projects, which have achieved maturity. Many projects do not pass the initial development stage. However, even successful projects experience periods of no change or little change.
II	Increasing complexity	Evidence is so far contradictory. There are some OSS systems that show increasing complexity and others of decreasing complexity. There is evidence of complexity control, but it is not clear how this affects the overall complexity trend. Structural complexity has many dimensions, and only a handful of them have been measured so far.
VI	Continuing growth	The law seems to describe well successful OSS where, despite irregularity in patterns, there is a tendency to grow in functionality. Some successful OSS systems like Linux display superlinear growth. However, many OSS projects also display none or little growth.
VII	Declining quality	This law is difficult to test because it depends on the measurement of quality. At least it should consider in addition to defect rates, the number of requirements waiting to be implemented at a given moment in time. These variables are difficult to study in OSS because, in general, there are no formal requirements documents.

[a]Abbreviation: OSS, open-source software.

projects, aiming to detect unmaintained projects and to develop a metric to alert developers about the risks of depending on a given GitHub project, while also attracting new collaborators to attractive projects. Rashid et al.[6] focused on the ever-changing and ever-transient nature of OSS project contributors (who are constantly joining, leaving, or changing their role) and the intrinsic loss of knowledge for the project that this has as a consequence. To mitigate such risk, they described the design and development of a robust research methodology in order to contribute towards the formation of proactive knowledge retention practices in OSS projects.

2.3 | Open software ecosystems

Research interest in software ecosystems and OSS has risen significantly in the recent years.[17] As more software developers include OSS packages into their projects, it also becomes relevant to study both the structure of open-source package ecosystems and the risks they convey, as Kikas et al.[18] did in their analysis of the dependency networks of the JavaScript, Ruby and Rust ecosystems. They found these ecosystems to be alive and growing, and having some popular packages being used in the majority of the projects, although they were becoming less dependent on a single popular package over time (thus reducing the risks of an ecosystem collapse). Such conclusions would, therefore, follow the laws of continuous growth and continuing change (being alive and mature implies adaptation) and show the ambiguity of Laws II and VII as both the reduction of complexity and the potential associated nondeclining quality could be explained by the community efforts. The influence of the CRAN ecosystem in particular on the scientific community has been recently examined, using a social network perspective, by Zanella and Liu.[19]

Another recent work by Decan et al.[20] empirically compared the evolution of the dependency networks in seven software packaging ecosystems, and proposed novel metrics to analyze and compare their evolution, considering their findings instrumental to assess and improve the quality of package ecosystems. In their future work section, they also pointed at the idea of empirically verifying Lehman's laws of software evolution in package ecosystems. Their work was derived from previous studies by the same researchers,[21,22] with two studies particularly focused in CRAN and the R language.[23,24] In the latter, the authors explored the ecosystem of software packages for R, not only in CRAN but also in other repositories (viz., BioConductor, R-Forge, and GitHub). They empirically explored the sizes of the repositories, the dependencies, and their growth, providing insight into the global evolution of the R package ecosystem back in 2015. They analyzed 12,000 packages divided among the four repositories (in comparison, at the end of 2019 CRAN has more than 15,000 packages) and concluded that CRAN was the central repository, mainly because its packages did not depend on external packages.

Before that, German et al.[25] conducted an exploratory empirical study on the evolution of the R software ecosystem, and showed how R was 'a flourishing ecosystem of user-contributed packages' that was growing and contained a 'strong set of core packages'. Among their observations, they found packages to be typically well-maintained (which could avoid falling in the decreasing quality law) and the number of dependencies per package to be 'consistently low accross all package sets' in 2013, 7 years before the current analysis.

As more developers collaborate in OSS ecosystems and reuse packages, while more final users depend on their correct operation, research on their structure and risks becomes relevant, as 'understanding the structure of software systems can provide useful insights into software engineering efforts and can potentially help the development of complex system models applicable to other domains'.[26] The evolution of the size, dependencies, or vulnerabilities (and the trends they follow) in CRAN, among other aspects, may be useful to developers and practitioners who focus on other OSS projects to, for instance, assess the health of the ecosystem.

In particular, the current study will explore the evolution of the CRAN ecosystem from the perspective of Lehman's laws in the last two decades, from its beginning in the late 90s to the end of 2019, complementing previous work on the laws of software evolution in the context of OSS and updating previous studies on the R ecosystem with more than 3 years of additional data.

3 | RESEARCH METHODOLOGY

3.1 | Data extraction

To extract the data, two methods were used and combined for the later analysis. The first extraction was executed using R and the 'pkgsearch' package,[27] which uses the 'R-hub' search server (see <https://r-pkg.org>) and the CRAN metadata database to provide detailed information about CRAN packages. The extracted metadata per package includes the following key elements among others:

- Descriptive features, such as name of the package, description, or version.
- Author(s).
- Imports: dependencies that are required for the package to work.
- Suggests: packages that can be used by the package but that are not required.

- Depends: currently states the version of R required by the package, but it is relevant as before R 2.14.0 this field contained the dependencies to other packages (therefore, it was equivalent to imports).
- Date/Publication: time stamp with the date of publication of the particular version of the package.

The extraction results in a total of 15,368 unique packages as of 31 December 2019. As many packages have multiple versions or releases, these 15,368 packages, in turn, become a total of 91,321 different versions being analyzed.

Although the extraction using the 'pkgsearch' package covers most of the features needed for the following analysis, data were manually cross-checked to ensure the reliability of the package and the obtained information was complemented with the information directly scrapped from the CRAN web repository at https://cran.r-project.org. The number of packages obtained using this method is exactly the same, 15,368, and the results were positive, so we can depart from the assumption that the information obtained from 'pkgsearch' is reliable.

3.2 | Data analysis

The resulting list of packages obtained from the previous step was processed in order to obtain a clean list of packages with all the required features. Packages listed in 'imports' and 'depends' were combined in a single dependency list. Release dates were also recorded for each version to allow for an snapshot of any particular point in time.

The processed data were used to empirically assess whether the evolution of CRAN adheres to each of the four laws that are directly related to software evolution, together with a look at the future prospects for the ecosystem.

With regard to continuous change, and considering R as an OSS project, which has achieved maturity, the first hypothesis is that CRAN is going to exhibit a high level of updates among its packages that are going to be continuously updated to keep up with the latest requirements and user needs and that older packages will still receive relevant support.

With the increase in use of R as a programming language and the expected growth of CRAN, the second hypothesis is that a certain degree of increasing complexity will be found. To measure complexity empirically, four measures will be used: first, the average number of direct imports that will show how many packages each package depends directly to work, assuming that more dependencies imply more complexity to keep the ecosystem healthy.[28] The three other measures will be related to transitive dependencies. Transitive dependencies, unlike direct dependencies, are hidden; removing a package that is transitively required by many others can impact a large portion of packages and, thus, becomes a risk for the ecosystem. First, the evolution of the ratio of transitive dependencies to direct dependencies will be tracked; second, the proportion of top-level packages by depth of their dependency tree at the end of 2019 will be assessed, and finally, the P-impact index will be used to study the evolution of the complexity of the directed network formed by the package dependencies in CRAN over the last 10 years.

Definition 1 The P-impact index of an ecosystem E at time t is the number of packages in E at time t that are transitively required by at least P% of all the packages in E.[20]

Continuing growth will be studied from the perspective of the absolute number of packages, expected to be continuously growing with new contributions, and from the perspective of dependencies, which, related to increasing complexity, are also expected to be growing. The law of decreasing quality will be assessed as a corollary of the previous three laws, with additional perspective on the policies of CRAN.

4 | RESULTS

4.1 | Continuous change

According to the law of continuous change, a 'system must be continually adapted else it becomes progressively less satisfactory in use'.[29] Therefore, maintenance and updates are unavoidable. In the case of CRAN, this is reflected in the number of updates (or versions) that packages receive over time. Figure 1 shows how the total number of monthly updates increases with time and does so at a higher rate than that of the new releases. In this case, a superlinear growth is a better fit than a linear one (R^2 = 0.94 for the quadratic curve, 0.88 for a linear one).

In particular, the monthly ratio of new updates versus new releases (Figure 2) remains regular, specially in the last decade: Every month, there are approximately 7.5 times as many updates as new releases. The time series analysis of the previous ratio over the last 8 years is shown in Figure 3, where neither trend nor seasonality is deemed as relevant by the augmented Dickey-Fuller test[30] (p value = 0.047); the series can, therefore, be considered stationary.

Even though the ratio of updates remains regular, it is also worth to drill further down and look at the distribution of these updates over individual packages. Figure 4 compares the proportion of packages that, for each year in the last decade, received either no updates, one or two, three

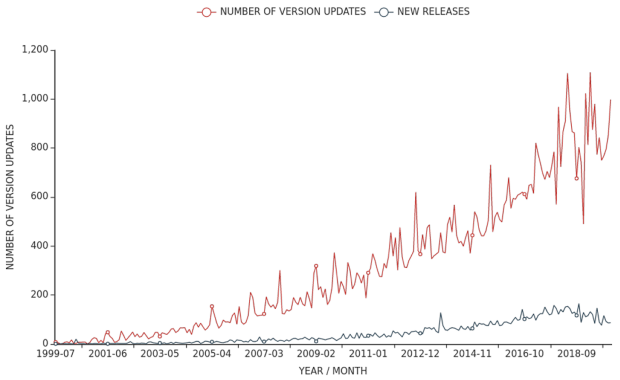

FIGURE 1 Monthly package updates compared to new releases

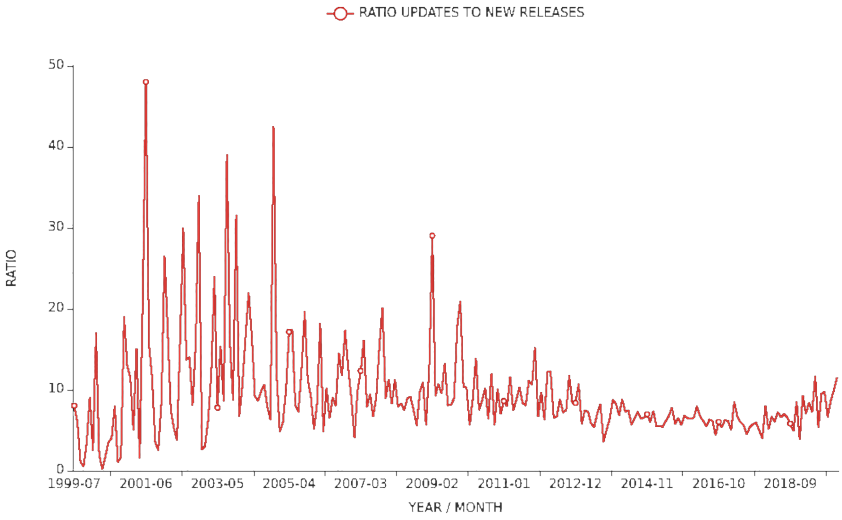

FIGURE 2 Ratio updates released to new packages monthly

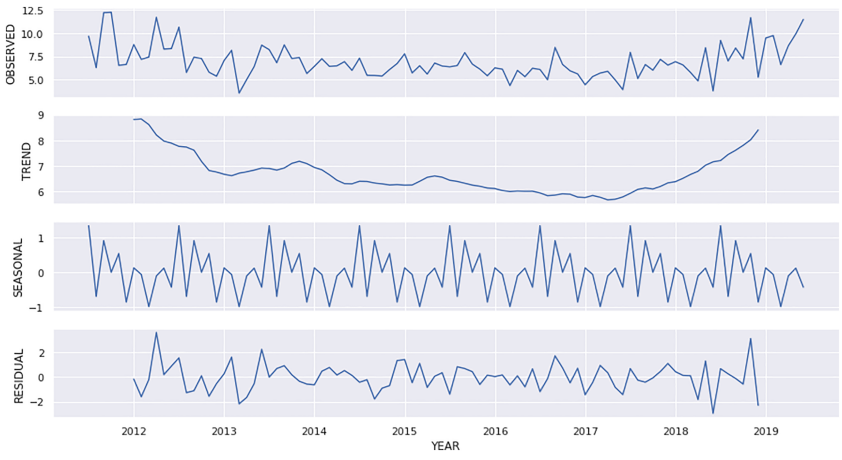

FIGURE 3 Time series analysis of the ratio of updates released versus the number of new packages

or four, or more than four updates. The four distinct bins correspond to an approximate equal proportion, about one quarter or 25% each, of the total number of packages of the ecosystem. It can be observed that most of the packages receive at least one update and that the average package receives at least three or four updates per year. These proportions are near to constant over time, showing how CRAN's packages are in continuous update.

Not only the update of CRAN's packages is continuous in time but it also is continuous regardless of package age. Figure 5 depicts the proportion of updates received in 2019 by all the available packages according to their age. Results reflect that most updates are received by packages older than a year; this is an evidence of both the age of the CRAN package repository and the continuous maintenance it gets in many of the oldest packages, which, in turn, are pivotal to many newer packages.

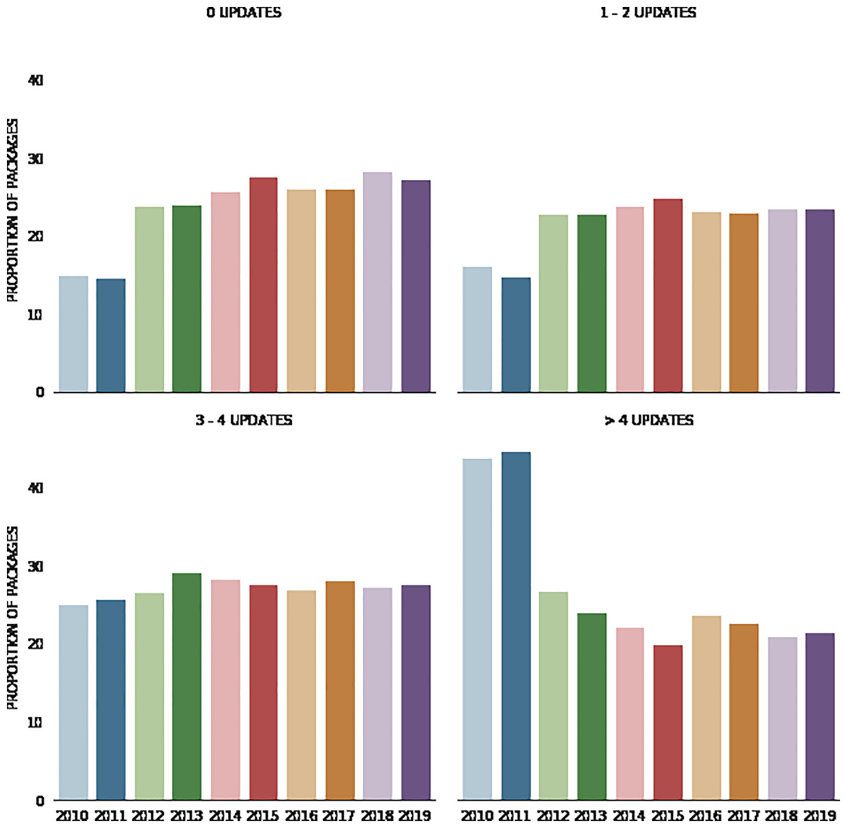

FIGURE 4 Proportion of packages having a certain number of yearly updates

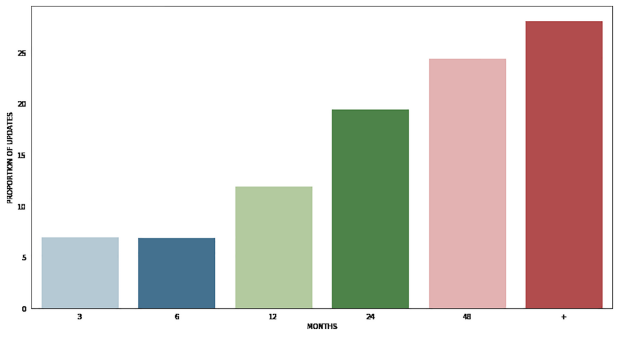

FIGURE 5 Proportion of updates in 2019 by age of the package (in months)

4.2 | Increasing complexity

In order to analyze the complexity of the CRAN ecosystem, measures related to the network of dependencies will be used.[31] The underlying principle is that as the number of dependencies increases (direct or, specially, transitive), the risk of failure is higher, as a broken package might affect more packages or break the ecosystem. In regard to direct dependencies (which, in CRAN, were first registered as 'depends' and later as 'imports'), Figure 6 reflects the rise in the average number of direct dependencies across new package releases and updates, exhibiting a notable growth in the recent years.

Additionally, the increase in the average number of dependencies does not translate into a concentration of dependencies in a few select packages, as the number of packages required to reach 80% of the total dependencies across the ecosystem has been growing, specially in direct dependencies in the last 5 years (Figure 7) (transitive dependencies also show higher figures but at a much lower rate than direct ones). Therefore, dependencies grow but do so among an also growing number of larger packages, introducing a higher number of potentially critical nodes for the ecosystem.

Decan et al.[20] suggested to use either the ratio of the number of dependencies over the number of packages or, as an alternative, the ratio of transitive over direct dependencies as a measure of the network complexity. The latter is represented in Figure 8, which also shows a relevant increase in the recent years and provides empirical evidence of the increase in complexity that the CRAN ecosystem is undergoing. The curve can be fitted to a quadratic growth with a R^2 of 0.993.

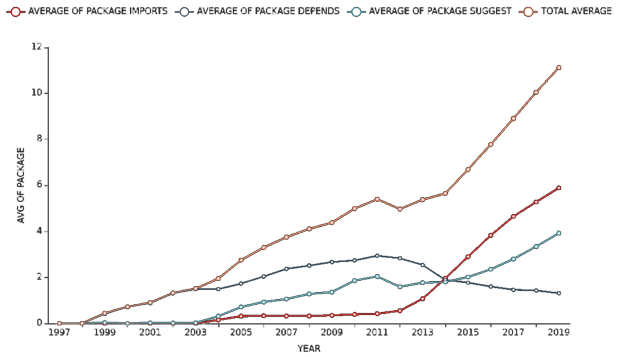

FIGURE 6 Average number of imports, depends, suggests, and total per year of publication

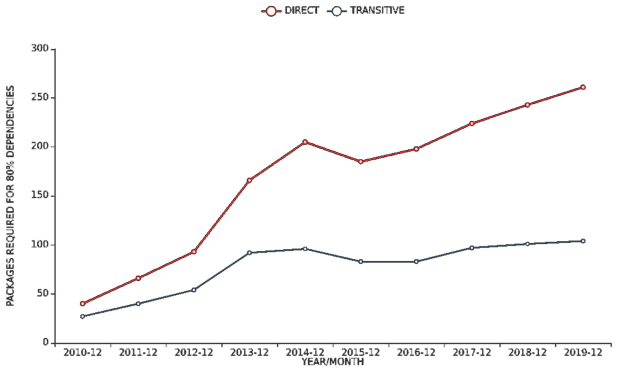

FIGURE 7 Evolution of the number of packages required to reach the 80% of the total dependencies

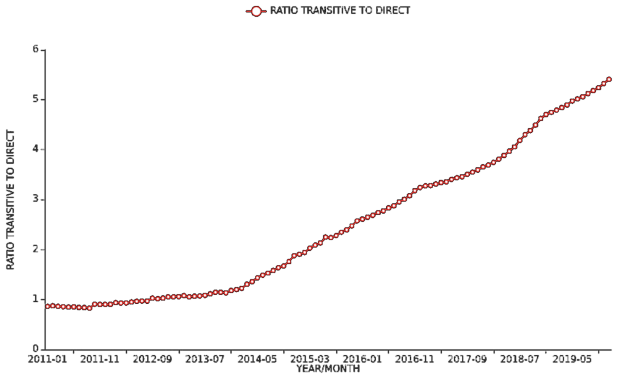

FIGURE 8 Ratio between transitive and direct dependencies over time

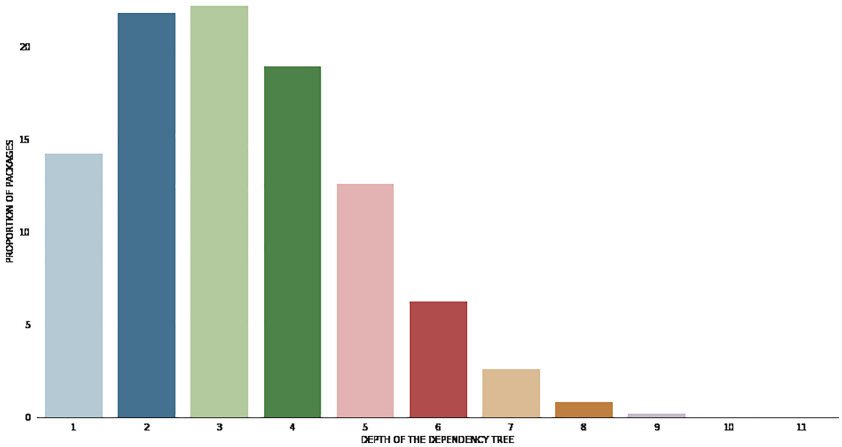

FIGURE 9 Proportion of top-level packages by depth of their dependency tree at the end of 2019

Related to the previous graphs, the depth of the dependency trees of the top level packages in CRAN (packages that depend on other packages but are not required themselves) is shown in Figure 9. The first observation is that an immense majority of the packages have a deep dependency tree; around two thirds of the packages are transitively dependent on others at least to level three.

The five-impact index (packages that are required directly or transitively by more than 5% of the total number of packages) shows an exponential increase during CRAN's life, but in the last 5 years, the fit for both the quadratic (R^2 = 0.981) and the linear models (R^2 = 0.986) is good and comparable. The 5% threshold was kept from the original metric authors;[20] in any case, the P-impact was also computed for other values (e.g., 2%), and similar results were obtained.

4.3 | Continuing growth

The hypothesis of continuing growth can be observed from two related perspectives: First, Figure 11 shows the cumulative number of packages in CRAN, which keeps growing at an approximate rate of more than 800 packages per year. The evolution over the last 15 years exhibits a super-linear growth (R^2 for the quadratic fit is 0.997 while a linear approach shows a smaller 0.888).

Second, it is also possible to look at growth from the perspective of direct dependencies (the edges of the package network), which, besides indicating pure growth of the ecosystem, also indicate a certain degree of additional complexity. In this case, the curve for the number of

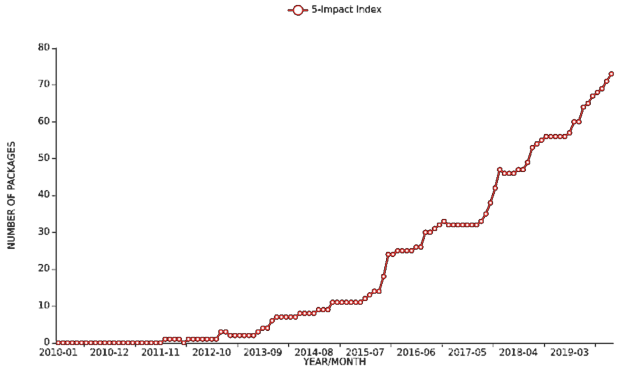

FIGURE 10 Five-impact index

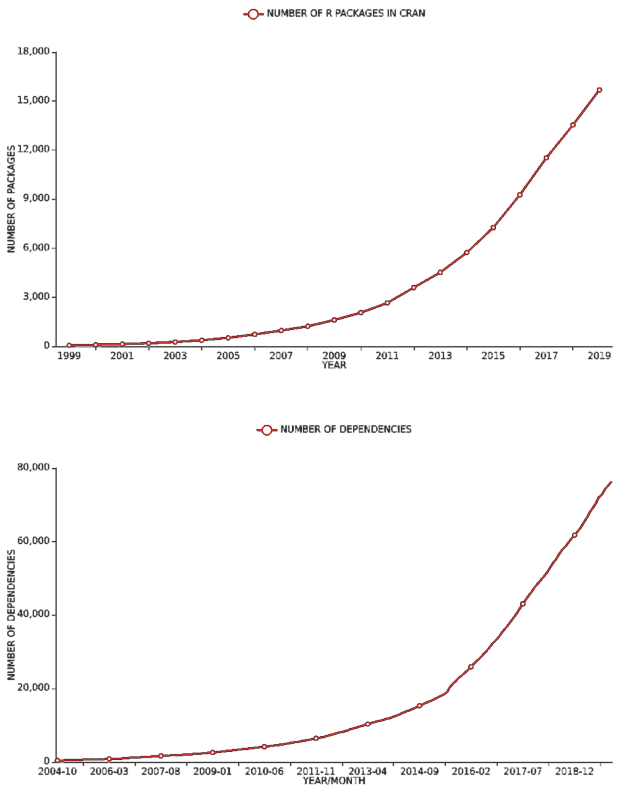

FIGURE 11 Number of packages available in Comprehensive R Archive Network (CRAN) per year

FIGURE 12 Number of dependencies from all packages in Comprehensive R Archive Network per month

dependencies (Figure 12) looks linear over the last 5 years, and a correlation test gives very similar results for both quadratic models (R^2 = 0.999) and linear models (R^2 = 0.997).

It is worth noting, however, that the monthly growth rate of these direct dependencies presents some deceleration in the last lustrum (Figure 13). The time series followed quite a regular constant growth in average until the end of 2015, when it started decreasing linearly (correlation shows an statistically relevant negative coefficient with goodness of fit R^2 = 0.72).

4.4 | Declining quality

The law of declining quality states that system quality declines unless it is actively maintained and adapted to environmental changes. Understanding this law as a corollary of the previous ones, it can be stated that, in CRAN, the continuous change reflected in, for example, the number of updates in packages (Figure 1) and the regularity on such updates (Figure 2) are part of the efforts done by the community to maintain and adapt the ecosystem in a constant fight against the decline in quality. Such efforts, however, might be hindered by the resulting increase in complexity, as seen in Figures 6 and 8 and, more importantly, in Figure 10, which shows how a large number of packages are already critical for a large portion of the package ecosystem.

In any case, confirmation of this law is hard to empirically test, as it depends on how quality is measured (and defined). In spite of this, it can be taken as anecdotal indication[14] that, as the adoption rate of R and CRAN has surged in recent years, its perceived quality is not decreasing but its usefulness is increasing instead, as the R ecosystem is not only increasing its use in its domain but also being used in environments outside what one would consider pure statistical computing, such as big data and data science projects.

It is also worth noting the existence of CRAN's strict policies on maintainers and contributions (https://cran.r-project.org/web/packages/policies.html). Among other policies, CRAN runs a periodic check on compatibility among packages; should any package fail the test, its maintainers would be notified and asked to resolve the issue before the following major R release, at the risk of having their package archived otherwise. CRAN also forces dependencies to be kept within itself (to avoid external dependencies). Additionally, backcompatibility versions of current packages is not allowed, and any changes to CRAN packages that could cause significant disruption to other packages must be agreed with the CRAN maintainers before releasing it. These policies have a direct impact in avoiding falling into declining quality, although it is at the cost of the CRAN's maintainers efforts.

4.5 | Future prospects

The present analysis used an ARIMA model,[32] to predict and forecast the evolution of the number of new releases (including new packages and updates to the existing ones) in the coming 5 years. The time series decomposition of the number of new releases is shown in Figure 14, with a clear growing trend and noticeable seasonality towards the start of each year, which results in a nonstationary result in the augmented Dickey-Fuller test. Thus, data are transformed using both a first difference and a seasonal first difference to comply with the assumption of stationarity before building the model.

The model predictions are available in Figure 15. It can be observed how the trend of notable increase in the last 5 years, where the average monthly releases has almost doubled from 507 in 2015 to 881 in 2019, is projected to keep increasing the number of releases, reaching a number close to 1,500 per month in 2024 that could potentially endanger the ecosystem unless there is an additional effort applied to its maintenance to avoid a decline on quality and usefulness.

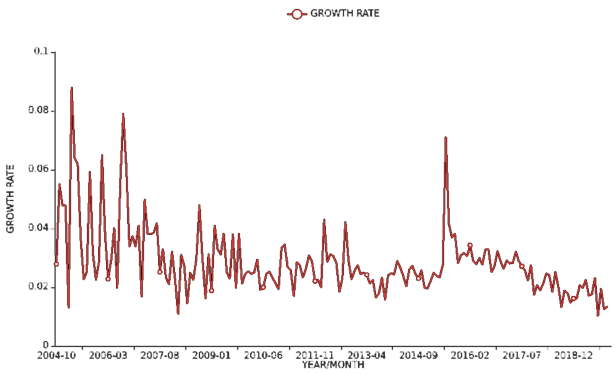

FIGURE 13 Monthly growth rate in number of direct package dependencies

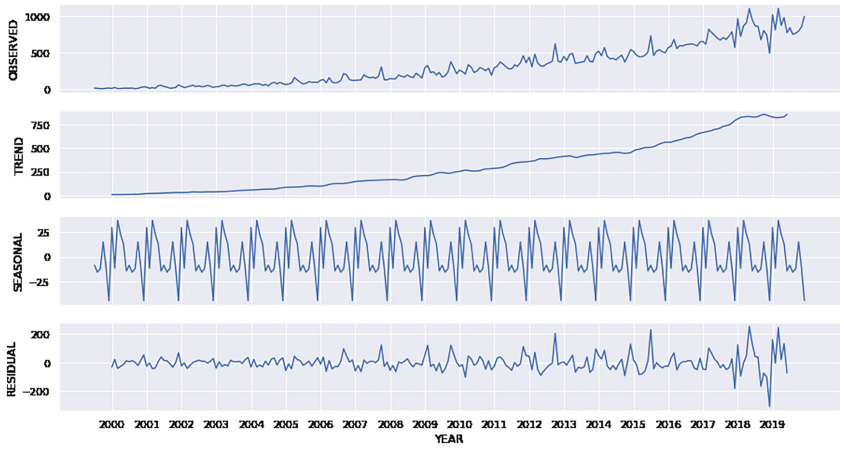

FIGURE 14 Time series decomposition of the number of new releases per month

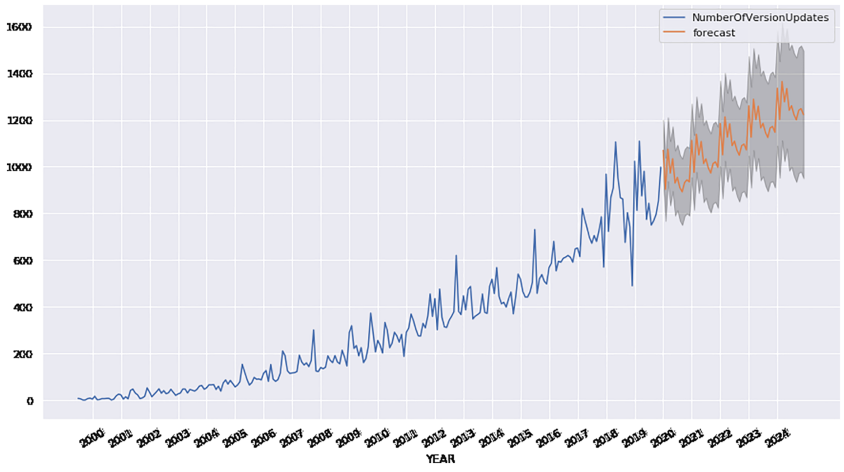

FIGURE 15 Forecast for the number of monthly new plus updated packages in CRAN

5 | DISCUSSION

This article has empirically examined the CRAN ecosystem, the main package repository for the R language, using the framework of Lehman's laws of software evolution. The first law, continuous change, seems to describe well how a CRAN has been evolving; although some irregularities might be found in the monthly patterns, there is an overall tendency to continuously updating the ecosystem and its functionalities. A large majority of the packages receive at least one update per year, with half of them receiving three or more. Such updates are specially focused on the older packages, that receive a higher level of attention, avoiding obsolescence and a potential risk for the newer packages that have them as dependencies.

In the same line, CRAN shows a clear tendency to a continuous growth in functionality. Similar to other OSS such as Linux,[14] the displayed growth both in absolute number of packages and in absolute number of dependencies is slightly over a linear growth. There are, however, some signs of deceleration in the growth of the ecosystem; even so, periods of none or little growth in the future would still be compatible with the results by other researchers.

The law of increasing complexity has been so far the most contradictory. In the current analysis, however, there are signs that CRAN has been experimenting a notable increase in complexity over the last lustrum. Not only the five-impact index and the average dependencies are growing in a superlinear tendency, but a comparison to the work of Decan et al.,[20] which contains data from 3 years earlier, results in significant differences. First, Figure 9 is now much more skewed to the right, with an increasing proportion of top-level packages having a deeper dependency tree, which introduces more risks to the ecosystem. Second, the ratio of transitive to direct dependencies was observed to be approximately stable over time in previous work, while the evidence in Figure 8 shows a relevant rise in recent years, similar to what happens with Figure 10. Therefore, the observed increase in complexity is compatible with Lehman's law, but it is also an indicator of a potential future risk unless mitigated by, for example, merging packages or functionalities, or additional efforts in reducing the depth of the transitive dependencies, as they could become an issue for the quality of the ecosystem.

The law of declining quality, in turn, has been related to the previous laws as it has been considered a corollary and it is also difficult to empirically prove; even more so when CRAN's policies are directly directed to the practice of preventative maintenance and avoiding a decline in quality, which, considering the surge in the use of R and CRAN, could be considered as effective. To keep the same level of usefulness, however, additional effort will be needed as next years could potentially represent a relevant increment in the number of packages and updates being published.

6 | THREATS TO VALIDITY

6.1 | External validity

Our work examined the CRAN package ecosystem. We recognize that our findings might not generalize to other OSS or commercial systems beyond those that have package networks with a similar structure to that of CRAN (i.e., package managers of other programming languages). We don not expect similar results in other kinds of package dependencies networks where, for instance, packages are more often installed by end users rather than being reused by developers in other packages. Some additional features could be used in the future (e.g., versioning) although in the case of CRAN, its own policy ends forcing compatibility with the latest version, and, on the other hand, the trends observed would not be expected to change. Additionally, although all the used features might not be available in other OSS projects, the most relevant ones (size, dependencies, and updates) are available in most repositories, which we intend to include in future research.

6.2 | Internal validity

As described in the research methodology, the data were extracted automatically from the CRAN metadata database through the 'pkgsearch' package in R. As there was no full guarantee of correct results from such tool, they were cross-checked with the information available in the CRAN web repository using a scrapping tool. Both results were consistent and, thus, deemed to be reliable. They were also compared with previous works where, with at least 3 years less of data, trends also appear to be consistent. The data snapshots on both sites were captured on 31 December 2019, which also needs to be taken into account in case packages had been removed before that date or in case the study is replicated at a later stage.

Additionally, the dependencies are measured according to the list of dependencies provided in each package metadata. In the case of CRAN, it must be noted that prior to the rollout of namespaces in R 2.14.0, the metadata field 'Depends' was the only way to reflect dependencies on another package. After that, developers are expected to use the field 'Imports' instead; to both account for older and newer packages, and possible inappropriate labeling in the metadata, both fields are combined to obtain the complete dependencies. This being said, however, notes how our analysis relies on what has been stated by each package developer or maintainer and some individual packages may have additional dependencies. In any case, we assume that such differences, if any, are irrelevant compared with the scale of the ecosystem.

Finally, the interpretation of results, specially when linking them to Lehman's laws of software evolution and due to their nature, may be subject to bias and different researchers might reach different observations or nuances.

7 | CONCLUSION

In summary, the evolution of the CRAN repository over the last two decades

- Has followed the law of continuous change, adapting to the needs of the community with regular updates on both newer and older packages, to ensure compatibility and usefulness.
- Has been growing in number of packages and links between them, following the law of continuous growth.
- There seems to be a recent and relevant increase in complexity that could represent a risk for the ecosystem in the coming years; transitive dependencies, specifically, could become a problem for the quality of the ecosystem as there is a large number of packages required transitively by a considerable portion of the whole ecosystem.
- With its continuous change, adaptation, and CRAN's policies (executed by CRAN's maintainers), declining quality seems to be avoided, at least from a usefulness perspective.
- CRAN is expected to keep growing in number of contributions over the next 5 years reaching up to 1,500 monthly new and updated packages, that could potentially overload the ecosystem.

7.1 | Implications for research and practice

Our research on the evolution of the CRAN package ecosystem, even with its exploratory nature, offers an analysis of one of the most popular OSS ecosystems as of today, used by researchers and practitioners alike. From a methodological standpoint, due to the longitudinal perspective over two decades and the exponential increase in collaborative development under the OSS paradigm, we complement the existing previous work with results that hint to a dangerous growth in complexity that should be carefully monitored over the next years in order to avoid falling into the decreasing quality trend described by Lehman's laws. The study also seeks to capture the complexity from the perspective of dependencies among packages; researchers on OSS ecosystems should not only keep their efforts on understanding their structures but also look for ways to reduce their complexity escalation via methodical reusage or integration of existing packages, for example. Furthermore, we show the interest of some of the existing metrics in the literature, which, in turn, calls for the development of further specific measures that could enhance OSS ecosystem research.

Although not in the scope of the present work, the results indirectly offer some suggestions or good practices for OSS package development. Developers would benefit from having a picture of the health of a given ecosystem and measures of the risks associated with, for example, particular packages with excessive dependencies or rarely updated. Such information could then be used by the developer or the development team to decide what functionalities to create or what parts to (safely) reuse from other packages, which would impact on the structure of the ecosystem and could, potentially, partially reduce its complexity while keeping a high level of adaptation and growth. The analysis could also be complemented with perspective from the contributors and the relationships between them and the evolution of the ecosystem. In summary, a detailed understanding of the factors that influence the success of the ecosystem would help contributors to the project in optimizing its design and maximizing the impact on the community, who would then benefit, in turn, of an optimized performance and usefulness.

7.2 | Future work

Three lines for future work are proposed. First and more critical, there is an urgent need to assess and understand what actions should and could be taken to mitigate CRAN's complexity and, therefore, to prevent future risks for the ecosystem. Second, the evolution of the ecosystem and its current state should also be analyzed from a complex network perspective, as complex network analysis can provide additional metrics, insights, and nuances that might not have been captured by the current work. Third, four of the eight laws of software evolution have been empirically checked in this article, but the other four have not; there is still ground for research in understanding the effect of the meta-laws and organizational laws in OSS and how to study them in decentralized user-contributed communities such as CRAN.

ORCID


Marçal Mora-Cantallops https://orcid.org/0000-0002-2480-1078

REFERENCES

1. Mens T, Claes M, Grosjean P. ECOS: Ecological studies of open source software ecosystems. In: *2014 Software Evolution Week - IEEE Conference on Software Maintenance Reengineering, and Reverse Engineering (CSMR-WCRE)*. Antwerp, Belgium: IEEE; 2014:403-406. https://doi.org/10.1109/csmr-wcre.2014.6747205
2. Lehman MM. Laws of software evolution revisited. *Software Process Technology*. Berlin, Heidelberg: Springer; 1996:108-124. https://doi.org/10.1007/bfb0017737
3. Gezici B, Tarhan A, Chouseinoglou O. Internal and external quality in the evolution of mobile software: an exploratory study in open-source market. *Info Softw Technol*. 2019;112:178-200. https://doi.org/10.1016/j.infsof.2019.04.002
4. de Vasconcelos JB, Kimble C, Carreteiro P, Álvaro R. The application of knowledge management to software evolution. *Int J Info Manag*. 2017;37(1):1499-1506. https://doi.org/10.1016/j.ijinfomgt.2016.05.005
5. Svahnberg M, Gorschek T. A model for assessing and re-assessing the value of software reuse. *J Softw Evol Process*. 2016;29(4):e1806. https://doi.org/10.1002/smr.1806
6. Rashid M, Clarke PM, O'Connor RV. A mechanism to explore proactive knowledge retention in open source software communities. *J Softw Evol Process*. 2019;32(3):e2198. https://doi.org/10.1002/smr.2198
7. Paulson JW, Succi G, Eberlein A. An empirical study of open-source and closed-source software products. *IEEE Trans Softw Eng*. 2004;30(4):246-256. https://doi.org/10.1109/tse.2004.1274044
8. Belady LA, Lehman MM. A model of large program development. *IBM Syst J*. 1976;15(3):225-252. https://doi.org/10.1147/sj.153.0225
9. Herraiz I, Rodriguez D, Robles G, Gonzalez-Barahona JM. The evolution of the laws of software evolution. *ACM Comput Surv*. 2013;46(2):1-28. https://doi.org/10.1145/2543581.2543595
10. Barry EJ, Kemerer CF, Slaughter SA. How software process automation affects software evolution: a longitudinal empirical analysis. *J Softw Mainten Evol Res Pract*. 2007;19(1):1-31. https://doi.org/10.1002/smr.342
11. Yu L, Mishra A. An empirical study of Lehman's law on software quality evolution. *Int J Softw Inform*. 2013;7(3):469-481.
12. Godfrey, Tu Q. Evolution in open source software: a case study. In: *Proceedings International Conference on Software Maintenance*. San Jose, CA, USA; 2000:131-142. https://doi.org/10.1109/icsm.2000.883030

13. Koch S. Software evolution in open source projects–a large-scale investigation. *J Softw Mainten Evol Res Pract*. 2007;19(6):361-382. https://doi.org/10.1002/smr.348
14. Israeli A, Feitelson DG. The Linux kernel as a case study in software evolution. *J Syst Softw*. 2010;83(3):485-501. https://doi.org/10.1016/j.jss.2009.09.042
15. Fernandez-Ramil J, Lozano A, Wermelinger M, Capiluppi A. Empirical studies of open source evolution. *Software Evolution*. Berlin, Heidelberg Springer; 2008:263-288. https://doi.org/10.1007/978-3-540-76440-3_11
16. Coelho J, Valente MT, Milen L, Silva LL. Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects. *Info Softw Technol*. 2020;122:106274. https://doi.org/10.1016/j.infsof.2020.106274
17. Serebrenik A, Mens T. Challenges in software ecosystems research. In: *Proceedings of the 2015 European Conference on Software Architecture Workshops - ECSAW 15*. Dubrovnik Cavtat, Croatia: ACM Press; 2015:1-6. 40. https://doi.org/10.1145/2797433.2797475
18. Kikas R, Gousios G, Dumas M, Pfahl D. Structure and evolution of package dependency networks. In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. Buenos Aires, Argentina: IEEE; 2017:102-112. https://doi.org/10.1109/msr.2017.55
19. Zanella G, Liu CZ. A social network perspective on the success of open source software: the case of R packages. In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*, Hawaii International Conference on System Sciences. Maui, Hawaii; 2020:471-480. https://doi.org/10.24251/hicss.2020.058
20. Decan A, Mens T, Grosjean P. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empir Softw Eng*. 2018;24(1):381-416. https://doi.org/10.1007/s10664-017-9589-y
21. Decan A, Mens T, Claes M. An empirical comparison of dependency issues in OSS packaging ecosystems. In: *2017 IEEE 24th International Conference on Software Analysis Evolution and Reengineering (SANER)*. Klagenfurt, Austria: IEEE; 2017:2-12. https://doi.org/10.1109/saner.2017.7884604
22. Decan A, Mens T, Claes M. On the topology of package dependency networks. In: *Proceedings of the 10th European Conference on Software Architecture Workshops - ECSAW 16*. Copenhagen, Denmark: ACM Press; 2016:1-4. 21. https://doi.org/10.1145/2993412.3003382
23. Decan A, Mens T, Claes M, Grosjean P. When GitHub meets CRAN: an analysis of inter-repository package dependency problems. In: *2016 IEEE 23rd International Conference on Software Analysis Evolution, and Reengineering (SANER)*. Suita, Osaka, Japan: IEEE; 2016:493-504. https://doi.org/10.1109/saner.2016.12
24. Decan A, Mens T, Claes M, Grosjean P. On the development and distribution of R packages. In: *Proceedings of the 2015 European Conference on Software Architecture Workshops - ECSAW 15*. Dubrovnik Cavtat, Croatia: ACM Press; 2015:1-6. 41. https://doi.org/10.1145/2797433.2797476
25. German DM, Adams B. The evolution of the R software ecosystem. In: *2013 17th European Conference on Software Maintenance and Reengineering*. Genova, Italy: IEEE; 2013: 243-252. https://doi.org/10.1109/csmr.2013.33
26. Zheng X, Zeng D, Li H, Wang F. Analyzing open-source software systems as complex networks. *Phys A Stat Mech Appl*. 2008;24(387):6190-6200. https://doi.org/10.1016/j.physa.2008.06.050
27. Csárdi G, Salmon M. Pkgsearch: search and query CRAN R packages. https://CRAN.R-project.org/package=pkgsearch; 2019.
28. Capiluppi A, Beecher K. Structural complexity and decay in FLOSS systems: an inter-repository study. In: *2009 13th European Conference on Software Maintenance and Reengineering*. Kaiserslautern, Germany: IEEE; 2009:169-178. https://doi.org/10.1109/csmr.2009.37
29. Lehman MM, Ramil JF. Rules and tools for software evolution planning and management. *Ann Softw Eng*. 2001;11(1):15-44. https://doi.org/10.1023/a:1012535017876
30. Cheung Y-W, Lai KS. Power of the augmented Dickey-Fuller test with information-based lag selection. *J Stat Comput Simul*. 1998;60(1):57-65. https://doi.org/10.1080/00949659808811871
31. Nikora AP, Munson JC. An approach to the measurement of software evolution. *J Softw Mainten Evol Res Pract*. 2005;17(1):65-91. https://doi.org/10.1002/smr.303
32. Hillmer SC, Tiao GC. An ARIMA-model-based approach to seasonal adjustment. *J Am Stat Assoc*. 1982;77(377):63-70. https://doi.org/10.1080/01621459.1982.10477767

How to cite this article: Mora-Cantallops M, Sicilia M, García-Barriocanal E, Sánchez-Alonso S. Evolution and prospects of the Comprehensive R Archive Network (CRAN) package ecosystem. *J Softw Evol Proc*. 2020;32:e2270. https://doi.org/10.1002/smr.2270