Journal of Information Technology Research
Volume 12 • Issue 4 • October-December 2019

An Empirical Study on the Network Model and the Online Knowledge Production Structure

Quan Chen, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan, China & School of Business Administration, South China University of Technology, Guangdong, China

Jiangtao Wang, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan, China

Ruiqiu Ou, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan, China

Sang-Bing Tsai, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan, China


https://orcid.org/0000-0001-6988-5829

ABSTRACT

Mass production has attracted much attention as a new approach to knowledge production. The R software system is a typical product of mass production. For its unique architecture, the R software system accurately recorded the natural process of knowledge propagation and inheritance. Thus, this article established a dynamic complex network model based on the derivative relationship between R software packages, which reflects the evolution process of online knowledge production structure in R software system, and studied the process of knowledge propagation and inheritance via the dynamic complex network analysis method. These results show that the network size increases with time, reflecting the tendency of R software to accelerate the accumulation of knowledge. The network density and network cohesion decrease with the increase of scale, indicating that the knowledge structure of R software presents a trend of expansion. The unique extension structure of R software provides a rich research foundation for the propagation of knowledge; thus, the results can provide us a new perspective for knowledge discovery and technological innovation.

KEYWORDS

Evolution Characteristics, Information Technology, Knowledge Production, Network Model, Production Structure

1. INTRODUCTION

The concept of mass production (peer production) is described as "the pattern of knowledge product production that is distributed together by the distributed users and jointly owned by the users." A typical feature of this new mode of production is that of non-central control: the producer voluntarily chooses the production content and result sharing, the producer is the user, and the output knowledge product is public (Benkler, 2006). All of these aspects are based on the Internet. The mass production project mode is mainly divided into 2 categories: Free Open Source Software Mode and Online

DOI: 10.4018/JITR.2019100109

Encyclopedia Mode. Open source software and online encyclopedias belong to the knowledge product area of online production, but the participants in the open source software need to have the ability to program. Because of this need for programming ability, the requirements for the participants are better than those for participants in online encyclopedias. The corresponding participation group is relatively stable, and the cooperation relationship between the knowledge producers in the community is more stable and persistent. In 2009, Black Duck reported that the cost of open source software development was estimated at 387 billion US dollars. Increasing numbers of software companies are involved in the development of open source software; one such example is that of Oracle's purchase of Sun's open source project for $7.4 billion in 2010. Google investment has created the open source community Google Code and the open source database MySQL. In the open source community, subsequent developers can create innovations based on the creations of earlier developers. This type of piggy-backing is a derivative of the anonymous cooperation model. It has not only greatly improved the efficiency of the creations of the developers but also promotes the development of knowledge products; that is, the speed of the development of open source software. The open source development mode provides a new way for the transformation of the industrial mode. Openness and transparency in the open source community can help to quickly gather public wisdom and effectively promote the formation and development of a new knowledge ecosystem.

Since it is a new approach to knowledge production, the mass production mode has attracted significant attention. Benkler and Nissenbaum (2006) examined how production cooperation can lead to knowledge innovation and communication in strange communities from an ethical perspective. The collaborative production mode in the open source community is usually described as a "virtual team" (Cohendet et al., 2001; Wellman, 1997). The research on the open source community can be summed up in 3 aspects: the participants' motivation for research, type of community the participants belong to, and network analysis of the community relations.

The research on the participants' motivations can be roughly classified into 3 categories: external motivation, intrinsic motivation, and internalized external motivation. External motivation includes career development (Hann et al., 2002; Hars & Ou, 2001; Orman, 2008; Hann et al., 2004), intrinsic motivation includes an interest in sharing (Ghosh, 1998) or learning opportunities (Shah, 2006; Ye & Kishida, 2003), and internalized motivation includes the development of the developer's own use requirements (Lakhani & Von Hippel, 2003; Lerner & Tirole, 2002). Henkel (2006) studied companies' participation in Linux open source community discovery and found that companies' desire to get external technology support was the main motivation for participation in open source. The motivation for individual participation is not only related to internal motivation (such as personal needs or prestige) but also related to the project community (such as leadership efficiency, interpersonal relationships, and community ecology) (Xu et al., 2009). Developers in different regions have different dominant motives (Subramanyam & Xia, 2008). Some studies also identified the motivation of dynamic participation (Shah, 2006; Wu et al., 2007).

The literature about the characteristics of community participants is mainly focused on the role characteristics of the participants in the community. For example, Mockus et al. (2000) indicated that the community had 3 types of contributors: source code providers, program error correctors, and error reporters. The number of contributors reporting problems was the greatest, error correctors came next, and the source code providers had the lowest number of participants. Only a few source contributors provided most of the source code, while the error correction amount of the program error correctors was relatively average. Ye and Kishida (2003) divided the open source community participants into 8 categories (project leader, core developer, developer, active developer, peripheral developer, error reporter, modifier, and reader) according to the core degree of the participant's position in the community. Crowston and Howison (2005) divided the participants in the open source community into 4 groups (founders and coordinators, developers, active users, and inactive users) from the perspective of management and compared the community structure to onions to reflect the

differences and hierarchical characteristics of the developer community. Crowston and Howison (2006) suggested that the participation of open source community members generally has an onion-type hierarchy, the core of which is the programmer; the outermost layer belongs to the error reporter. The transition from outer layer to inner layer involves a high cost, and the number of personnel in the core layer is very small. Barcellini et al. (2009) divided the role of the community into 4 categories: the first category is the project leader, generally the initiator of the project; the second category is the project manager or the core developer, who is responsible for the maintenance of the code database and the files; the third is the initiator, and this person is involved in the process of software improvement; and the fourth category is users, including contributing users.

The literature on the relationships in the community focuses on the analysis of the collaborative structure characteristics in the community. For example, Hunt and Johnson (2002) studied the activity distribution of about 4,000 projects in the SourceForge community. The activity of these projects follows the Pareto distribution, which the authors attributed to "the winner takes all" attitude in the software production process. Madey et al. (2002) studied the cooperative network discovery in the open source community of SourceForge and found that of the total number of participants, the number of participants actually involved in the project and size of the connected graph all followed the power law distribution. Therefore, the power law distribution is 1 piece of evidence for the self-organizing production mode of the open source community. Gao et al. (2003) studied the network structure of 50,000 projects in the SourceForge open source community by using the dynamic 2-division network. The study found that the degree of the nodes and the size of the connected graph in the community were all subordinate to the power law distribution, with an aggregation coefficient of 0.7. Xu (2007) and his colleagues (2006) also found power-law distribution characteristics and small-world characteristics when studying the SourceForge open source community cooperation network. Stol and Babar (2009) gave a comprehensive summary and analysis of researches on open source software. Wang et al. (2012) established a cooperative social network through the use of developer partnerships and explored the impact of social networks on project performance in the open source community.

Although literature studied the mass production from various perspectives, few papers paid attention to knowledge production structure and its evolution process in mass productions. To fill this gap, we take R software system for example to study the evolution process of online knowledge production structure in mass production. R software is a typical knowledge product that was developed in the mass production mode. R software relies on its modularity, software package system, and eco-friendly environmental characteristics, enabling it to develop rapidly as open source software (Koch, 2005). The unique architecture of the R software package mode (Fox, 2009) makes the R software system more like a naturally growing organic software body. The system allows other developers to develop software packages independently without the intervention of the core developers of the software and allows any individual to exploit the new software package derived from the predominant feature code of the previous software package without the need for the conformance vote of other types of open source software. These software packages provide not only the process of knowledge dissemination and inheritance but also the process of human beings operating in the field of scientific exploration and engineering, on the basis of the continuous accumulation of previous algorithms and programming innovations.

Specifically, in this paper, we use dynamic complex network model to study the production process of R software system and reveal its evolution process of online knowledge production structure. The basic structure of this paper is as follows: the second section describes the extension process of the R software package. The third section uses the dynamic complex network model to describe the evolution characteristics of the derivative relationship of the R software package. The fourth section studies the expansion characteristics of the knowledge production structure of the software package. The final section summarizes our study.

2. R SOFTWARE

R software became open source in 1997 and is one of the most important tools in data science. The typical characteristics of R software are its interface, interactions, functions, object-oriented programming language, modularity, and global open source collaboration. In the list of programming software released by TIOBE in May 2013, R software surpassed SAS software for its use in statistics (TIOBE, 2013)[1]. At present, R software has more than 4,600 software packages and more than 6,700 official registered developers. The unique architecture of the R software package enables developers to use the functions of the existing software packages to reduce technical difficulties and technical barriers to development when developing new software packages with their own unique technology and ideas (Fox, 2009). New software is not subject to the intervention of the core developer of R software. Therefore, the developer only needs to use the advantages and characteristic codes of the previous software packages and reintegrate them into new functional software packages. The growth and utilization of R software is a naturally-growing organic software body. Different software package management systems reflect different software development control ideas and evolve different development paths.

The R software package system defines 3 relationships to organize and coordinate its development: (1) the dependency relationship, that is, the new software package calls the existing software package, then the new software package must be declared in the description file (description.txt) to automatically load the dependent software package at runtime; (2) importing the relationship, that is, importing the relationships through the introduction of the name space (Name Space) coordinates and the conflicts of the same functions between different packages; and (3) recommended relationships (suggests), which recommends that the developer use multiple software packages in the completion of complex problems. Dependency is the motivating force behind R software expansion, and the import relationship only solves the conflict problem of the name of the developer's function package. The recommendation relationship helps the software users make better decisions.

The software packages developed using R software contains unique knowledge. The technological background of this knowledge is accumulated knowledge and knowledge involved in solving the natural and social problems during scientific research. These knowledge structures are combined by programming technology, forming the knowledge structure of R software. In the R software package, there are substantial dependencies and recommendation relationships. R software is an organic combination of a large number of software packages. This organic combination of its internal structure provides a rich foundation for network analysis.

3. METHOD

The study was reviewed and approved by an institutional review board Zhongshan Institute, University of Electronic Science and Technology of China (ethics committee).

The resources of R software are stored in the R software master station (r-project.org), the package software's storage and downloading station CRAN (cran.r-project.or) and its mirrors around the world, the official Developer Platform (r-forge.r-project.org), and other important developer platforms (such as bioconductor.org and omegahat.org). In particular, the software package storage and download station CRAN contains authoritative and comprehensive package data. Therefore, the data of the software package mainly comes from CRAN. The data mining method from the software package description file of CRAN includes the name of the software package, earliest release time of the software package, and name of the software package that the software package depends on. It has collected 4,712 software packages and 10,868 groups of dependencies, of which 837 software packages do not have dependencies. We obtained effective data for 3,875 software packages and 10,868 sets of dependency relations.

4. DISCUSS

The dependency structure between the R software packages reflects the process of R software's extension and the derivative relationship between the software packages, that is, the knowledge propagation and inheritance. Therefore, based on the derivative relationship between the software packages, we constructed the network structure of the software package and described the dynamic network description of the dynamic network based on the time-associated edge. The time section is shown in Figure 1.

4.1. The Speed of Increase in the Number of Software Packages

In reality, the scale of the network has been changing dynamically. Figure 2 shows the function image of the number of nodes. The number of nodes increases with time, and the growth rate is increasing. The nodes in the R software package network correspond to the software packages. The growth of the nodes means the growth of the number of software packages, that is, that new software packages are being developed. The emergence of the new software package means the entry of new technologies, as well as the enrichment and extension of its peripheral applications. The right fitting of Figure 2 shows that the growth of the R software network shows a power law growth mode, with a power index of 0.165. The growth of the R software package is consistent with the expansion of its application scope. Since R software has rapidly gained popularity in scientific computing, it has been intently sought after by those in that industry due to the increasing demands to utilize big data. The number of people who have joined the R software development team has grown rapidly, and the number of R software packages has been growing exponentially in accordance with the power law.

Figure 1. Representative sections of the R software dynamic-directed network

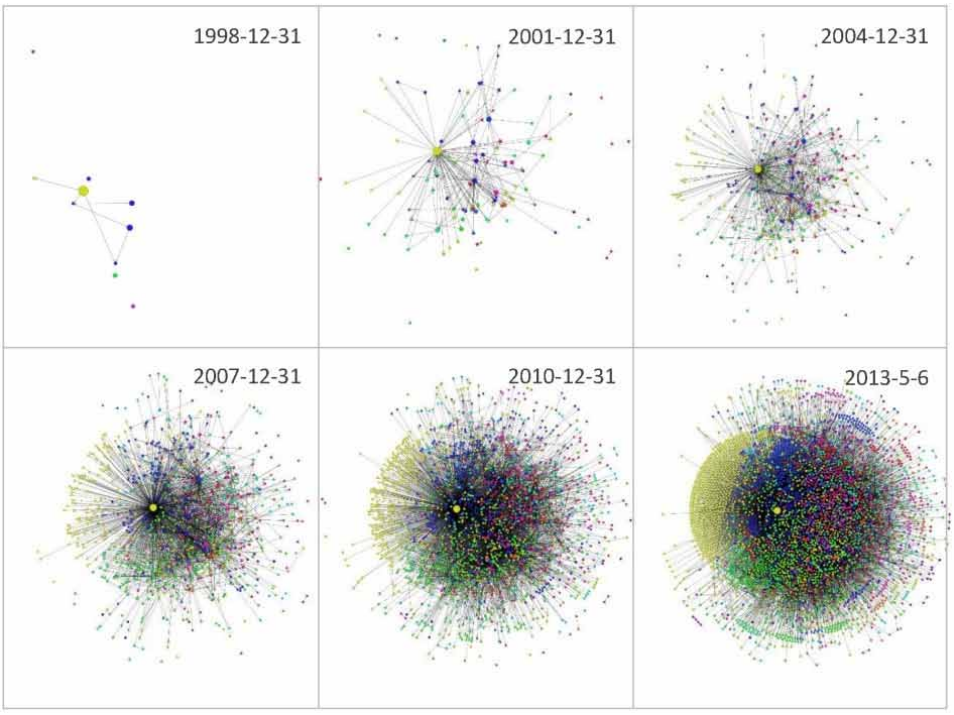
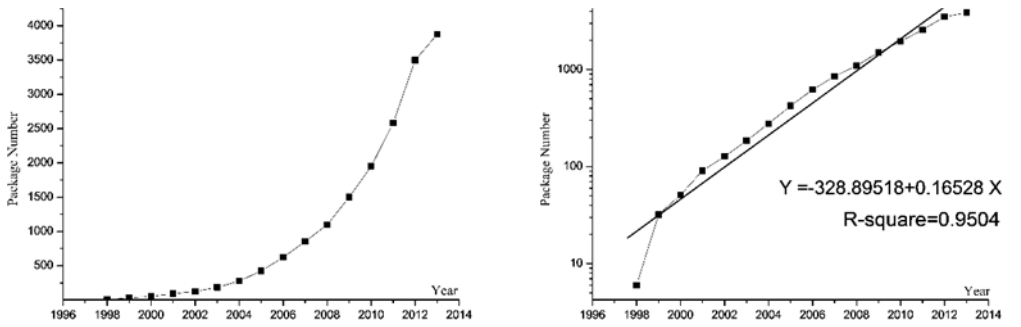


Figure 2. The size of the R software network



4.2. The Evolution Order of the Network Density

There are 2 ways to measure network density: One is density and the other is via the average degree. Density is defined as the number of connected nodes of a network divided by the maximum possible number of connected edges, and the average degree is defined as the average number of the connected sides of the nodes. The density is measured by the relative density of the network, and the average degree is the absolute density of the network. In the R software package on the network, the density and the average degree of *K* are a function of time, and these can be respectively denoted as $P(T)$ and $K(t)$. Figure 3 shows the $P(T)$ and $K(T)$ trends. The average output of R software has been stable at 2.7-2.8 since 2009, that is, on average, 2.7-2.8 new software packages can be derived from every software package. As the average degree of departure tends to be constant, the network density tends to be sparse as the number of nodes N increases.

4.3. The Derivative Speed of the Software Package

The increase in the relationship between the software packages is derived from 2 factors: first, the integration and innovation between the old packages, and second, the direct derivation and innovation of single software packages. The occurrence of the former includes multiple derivative relations and the latter contains a relationship.

Figure 4 shows the growth trend of the R software network variables. If $M(T)$ and $N(T)$ are the number of edges and nodes that change with time, respectively, it is known from the relation formula of the density that when N goes to infinity, the network density p tends to a constant, that is, M and N^2 are in the same order, so the network is dense. If the density tends to 0 when N goes

Figure 3. The evolution of the density (left) and mean degree (right) of the R software network

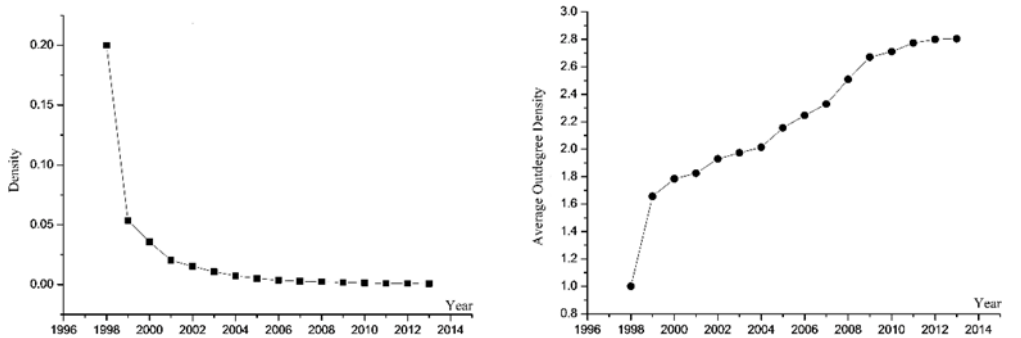
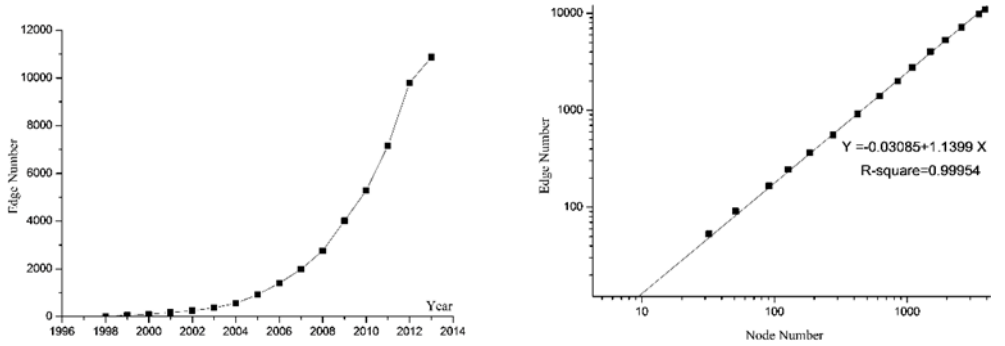


Figure 4. The super-linear growth relation of the number of edges of the R software network

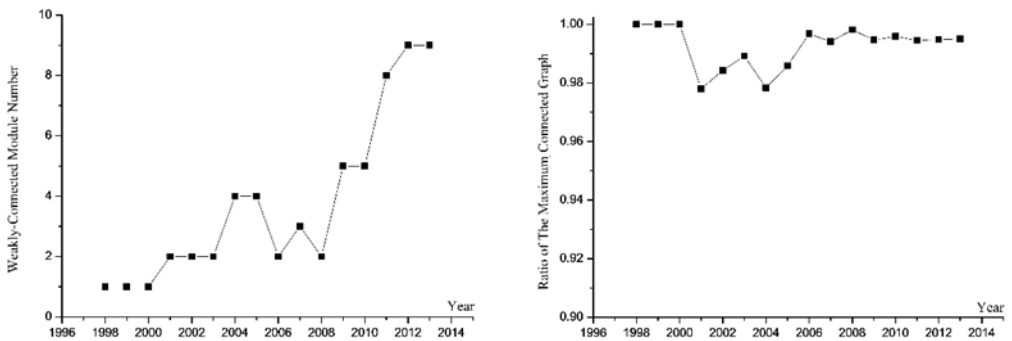


to infinity, M is lower than N^2, so it is sparse (Wang et al., 2012). The relation between fitting M (T) and N (T) in Figure 4 was found to be ln $M(T) = -0.03 + l.14 \ln N(T)$, i.e., $M(t) \sim N(t)^{1.14}$. Obviously, this relation also explains the increase of the average degree and decrease of the density in Figure 3. Due to the logarithmic growth of the number of nodes and the number of edges, the slope of the relation is also called the dense power law index.

4.4. Connected Modules

The number of connected modules is used to measure the connectivity of a network. The more connected the modules are, the worse the connectivity of the networks is. The larger the scale of the largest connected module is, the better the connectivity of the network is. The scale of the largest connected module takes into account the connectivity of the network in a relative sense, and the size of the network connects the connectivity of the network in an absolute sense. For the R software network, the connectivity represents the continuity of the R software development. Other software packages that provide an interface to R software may not rely on the R software package, but they will still be dependent on the newly developed R packages, resulting in the fragmentation and dis-connectivity of the whole network. Because of the characteristics of the R tree growth, there is no strong connected structure similar to the Internet. For this reason, we studied the connectivity of R from the perspectives of the weakly-connected graphs, number of connected graphs, number of software packages, and ratio of the maximum connectivity graph in Figure 5.

Figure 5. The evolution of connectivity: The number of connected modules (left) and the maximum number of nodes connected to each other (right)



The number of weakly-connected modules increased from 1 to 9. The scale of the largest connected module was above 97%. The above phenomenon also indicated that the R package network had a good connectivity. Further research showed that the connectivity of the R software network increased between 1998 and 2013 and shows a very stable trend. Figure 6 shows that R software is different from other types of open source software due to its stable and unbranching development of the main body. Its method of development results from the fact that the largest connected giant is constantly absorbing and gathering free small connected segments. Among them, the largest connectivity occurs in the main body of the R development, so the total package number was basically stable at the 99.5% level. In addition, the increase in the other connected modules meant that the compatibility and integration of other software packages for R is increasing. In reality, commercial statistics, data mining, and large data software actively integrate R, which makes the connectivity of the peripheral R more likely to increase.

4.5. The Evolution Order of the Agglomerate

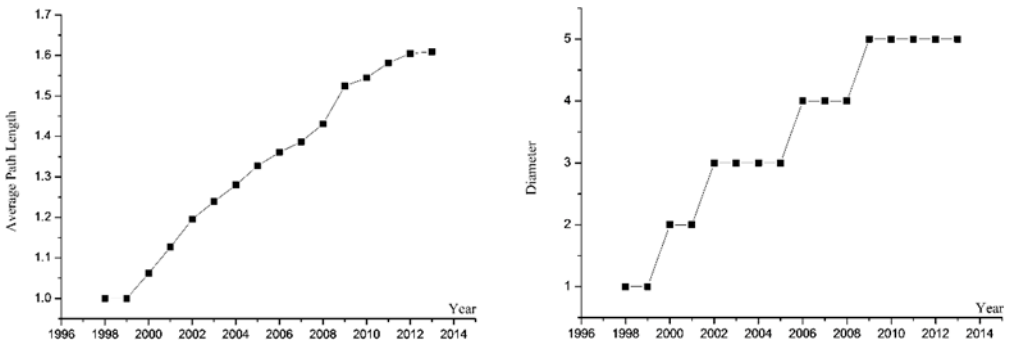
Based on the extension of knowledge structure of R software, the cohesion between R packages is measured by the average path length and network diameter indexes.

R is a purely statistical early programming language. It does not include graphical interfaces or its own KDE tools and is not friendly to new users. General statistical data processing was often unable to satisfy all the subjects' applications, so the application groups gradually expanded. The parties with needs extended the new algorithm and new model on the basis of the general packet. Subsequent developers can choose to join the original package author team to add their new achievements or extend the original packages to form new packages. From the length of the extension of the software package chain, a growth mode of R software may eliminate the distance between packages. However, newly generated packages can also continue to depend on the basic general statistical methods. From the point of view of the network model, there is a long-range edge that causes the average path length to be shorter. The growth mode of a software package reduces the growth rate of the global average path length, even though the absolute average path length is increasing. Due to the addition of a dependency packet on the longest geodesic line, it is possible to increase the diameter by 1. Beginning in 2009, the diameter of the R software network was stable at a level 5.

4.6. The Evolution Order of Heterogeneity

Network heterogeneity refers to the heterogeneity of the nodes, which occurs due to the uneven distribution of the degree values. The heterogeneity of the degree distribution is one of the most important discoveries in complex networks since 2009 (Barabási, 2009). The R software network is a spontaneous growth network. Its heterogeneity is related to whether there is a spindle in the software's growth, which may lead to bifurcation. The relatively high heterogeneity in the R software

Figure 6. The evolution of the average path length (left) and diameter (right) of the R software network



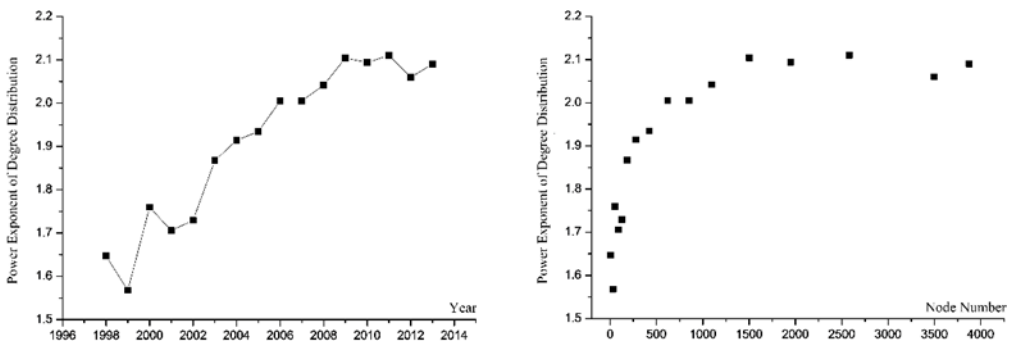
network corresponds to the relative concentration of derived relationships, which means that the development content is relatively stable and focused. Through fitting the degree sequence of the R software network over the years, we obtained the index evolution result shown in Figure 7. The power exponent distribution of the R software network was between (1.5,2.1), which is a typical II type heterogeneity. Therefore, the development of R software is focused on some nodes, and the main axis is distinct. According to the process of the R software network node scale expansion, the scale of R software is still expanding rapidly, but the power exponent is basically stable at about 2, maintaining the characteristics of the type II heterogeneity.

5. CONCLUSION

Mass production has attracted much attention as a new approach to knowledge production. Open source software, a typical application of mass production, is a very successful large-scale collaborative practice of the mass production mode in software on the Internet. The success of open source projects not only includes new software development methods but also changes the pattern of the software industry. The Internet-based knowledge production model has subverted the traditional business model. The unique extension structure of R software provides a rich research foundation for the propagation of knowledge. Therefore, in this study, we examined the dynamic evolution characteristics of a knowledge production structure by virtue of the specific structure system of the R software package and the dependence of the relationship between its software packages. Our results show that the network size increases with time, reflecting the tendency of R software to accelerate the accumulation of knowledge. The network density decreases with the increase of scale, and the network cohesion decreases with the increase of scale, indicating that the knowledge structure of R software presents a trend of expansion. The network connectivity is good. The proportion of the largest connected giant pass was more than 97%. The development mode of R software shows that the main body of the largest connected giant is continuously absorbing and gathering free, small connected segments. Finally, the network has a small number of nodes, which reveals that the main axis of the knowledge structure of R software is not bifurcated.

As we known, the division of labor and collaboration are the nature of market economy. The increase of economy scale is characterized by deepening of division of labor and expansion of collaboration. As Hayek pointed out, the essence of division of labor is division of knowledge. In this point of view, the booming of internet tremendously promotes the deepening of division of labor, which further promotes the economy growth. Nowadays, many information technology firms, such as IBM, Google, Facebook, Huawei and Xiaomi, deeply participate in online collaborative production activities under the open-source setting. Therefore, it has practical significance to investigate the division of labor and collaboration in online knowledge production. This paper studied the knowledge

Figure 7. The evolution of the power exponent and power exponent of R software with scale



structure of the R software system's online production, and revealed evolution characteristics of this knowledge structure from several aspects. The results may help firms improve the efficiency of online software developments and of online open innovation.

ACKNOWLEDGMENT

This research was supported by the National Natural Science Foundation of China (71301054), the fundamental research funds for the central universities (2015QNXM13, 2017X2D14) and Provincial Nature Science Foundation of Guangdong (2015A030310271, 2015A030313679, 2015A030313681) and Zhongshan City Science and Technology Bureau Project (No. 2017B1015) and 2018 Zhongshan Innovation and Development Research Center and Guangdong Academy of Education Project (GDJY-2015-C-b010).

REFERENCES

Barabási, A.-L. (2009). Scale-free networks: A decade and beyond. *Science*, *325*(5939), 412–413. doi:10.1126/science.1173299 PMID:19628854

Barcellini, F., Détienne, F., & Burkhardt, J. M. (2009). Participation in online interaction spaces: Design-use mediation in an open source software community. *International Journal of Industrial Ergonomics*, *39*(3), 533–540. doi:10.1016/j.ergon.2008.10.013

Benkler, Y. (2006). *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.

Benkler, Y., & Nissenbaum, H. (2006). Commons-based peer production and virtue. *Journal of Political Philosophy*, *14*(4), 394–419. doi:10.1111/j.1467-9760.2006.00235.x

Cohendet, P., Creplet, F., Dupouët, O. (2001). Organisational Innovation, Communities of Practice and Epistemic Communities: the Case of Linux. *Economics with Heterogeneous Interacting Agents*, (51), 303-326.

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, *10*(2), 405–411. doi:10.5210/fm.v10i2.1207

Crowston, K., & Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology & Policy*, *18*(4), 65–85. doi:10.1007/s12130-006-1004-8

Fox, J. (2009). Aspects of the social organization and trajectory of the r project. *The R Journal*, *1*(2), 5–13.

Gao, Y., Freeh, V., & Madey, G. (2003). Analysis and modeling of open source software community. *Naacsos*, *1*(1), 1–12.

Ghosh, R. A. (1998). Interviews with linus torvalds: What motivates software developers. *First Monday*, *3*(2), 1–12. doi:10.5210/fm.v3i2.583

Hann, I. H., Roberts, J. A., & Slaughter, S. (2004). Why Developers Participate in Open Source Software Projects: An Empirical Investigation. In *International Conference on Information Systems ICIS 2004*, Washington DC, December 12-15 (pp. 821-830). DBLP.

Hann, I. H., Roberts, J. A., Slaughter, S., & Fielding, R. (2002). Economic Incentives for Participating in Open Source Software Projects. In *International Conference on Information Systems ICIS 2002*, Barcelona, Spain (p. 33). DBLP.

Hars, A., & Ou, S. (2001). Working for free? Motivations of participating in open source projects. In *Hawaii International Conference on System Sciences*. IEEE.

Henkel, J. (2006). Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy*, *35*(7), 953–969. doi:10.1016/j.respol.2006.04.010

Hunt, F., & Johnson, P. (2002). On the pareto distribution of sourceforge projects. In *Open Source Software Development Workshop*.

Koch, S. (2005). *Free/Open Source Software Development*.

Lakhani, K. R., & Von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, *32*(6), 923–943. doi:10.1016/S0048-7333(02)00095-1

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, *50*(2), 197–234. doi:10.1111/1467-6451.00174

Madey, G., Freeh, V., & Tynan, R. (2002). The Open source software development phenomenon: An analysis based on social network theory.

Mockus, A., Fielding, R. T., & Herbsleb, J. (2000). A Case Study of Open Source Software Development: *The Apache Server. In International Conference on Software Engineering* (Vol. 11, pp. 263-272). IEEE. doi:10.1145/337180.337209

Orman, W. H. (2008). Giving it away for free? the nature of job-market signaling by open-source software developers. *The B.E. Journal of Economic Analysis & Policy*, *8*(1), 1875–1875.

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*(7), 1000–1014. doi:10.1287/mnsc.1060.0553

Stol, K. J., & Babar, M. A. (2009). *Reporting Empirical Research in Open Source Software: The State of Practice. Open Source Ecosystems: Diverse Communities Interacting*. Springer Berlin Heidelberg.

Subramanyam, R., & Xia, M. (2008). Free/libre open source software development in developing and developed countries: A conceptual framework with an exploratory study. *Decision Support Systems*, *46*(1), 173–186. doi:10.1016/j.dss.2008.06.006

Wang, J., Hu, M. Y., & Shanker, M. (2012). Human agency, social networks, and foss project success. *Journal of Business Research*, *65*(7), 977–984. doi:10.1016/j.jbusres.2011.04.014

Wellman, B. (1997). *An electronic group is virtually a social network*. Kiesler S Culture of the Internet.

Wu, C. G., Gerlach, J. H., & Young, C. E. (2007). An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management*, *44*(3), 253–262. doi:10.1016/j.im.2006.12.006

Xu, B., Jones, D. R., & Shao, B. (2009). *Volunteers' involvement in online community based software development*. Elsevier Science Publishers B. V. doi:10.1016/j.im.2008.12.005

Xu, J. (2007). *Mining and modeling the open source software community*.

Ye, Y., & Kishida, K. (2003). Toward an understanding of the motivation Open Source Software developers. In *International Conference on Software Engineering* 2003 *Proceedings* (pp. 419-429). IEEE.

Ye, Y., & Kishida, K. (2003). Toward an understanding of the motivation Open Source Software developers. In *International Conference on Software Engineering* 2003 *Proceedings* (pp.419-429). IEEE.

Sang-Bing Tsai is currently the Professor and the Lecture Professor at University of Electronic Science and Technology of China Zhongshan Institute & Civil Aviation University of China. He is both Technology Management and Business Management PhD. He has published many research articles in SCI/SSCI journals. His recent research interests in Operation Management, Computer Science, Service Management, Financial Technology, Big Data, Sustainability and Applied Mathematics. He had much academic experience and industry experience. He is the IGI Global book series editor(s)-in-chief: Advances in Environmental Engineering and Green Technologies. He is at these international journal editorial teams as a editor: Journal of Global Information Management (Associate Editor) (SSCI); Sustainability (Lead Guest Editor) (SSCI/SCI); Sustainability (Editor) (SSCI/SCI); SpringerPlus (Editor) (SCI); International Journal of Digital Crime and Forensics (Associate Editor) (EI); International Journal of Knowledge Management (International Advisory Board Editor) (EI); International Journal of Environmental Research and Public Health (Lead Guest Editor) (SCI/SSCI). [1] TIOBE. TIOBE Programming Community Index for May 2013. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.