RAND Journal of Economics
Vol. 42, No. 1, Spring 2011
pp. 70–91

Direct and indirect knowledge spillovers: the "social network" of open-source projects

Chaim Fershtman*

and

Neil Gandal**

Knowledge spillovers are a central part of knowledge accumulation. The article focuses on spillovers that occur through the interaction between different researchers or developers who collaborate on different research projects. The article distinguishes between project spillovers and contributors' spillovers and between direct and indirect spillovers. The article constructs a unique data set of open source software projects. The data identify the contributors who work on each project and thus enable us to construct a two-mode network: a project network and a contributor network. The article demonstrates that the structure of these networks is associated with project success and that there is a positive association between project closeness centrality and project success. This suggests the existence of both direct and indirect project knowledge spillovers. We find no evidence for any association between contributor closeness centrality and project success, suggesting that contributor spillovers play a lesser role in project success.

1. Introduction

■ Knowledge spillovers facilitate the transfer of knowledge and ideas between firms, researchers, and research teams. The transmission of knowledge can be done in different ways. Individuals may learn by being exposed to or by studying different innovations or new technologies without any personal interaction with their developers. A second type of spillover occurs through interaction between individual researchers who discuss and exchange ideas and information. Academic research may provide a good example for these two types of spillovers. One may learn new ideas by reading and studying papers and academic research without any personal interaction with authors of the papers. The second form of knowledge spillover is by interacting with coauthors and colleagues.

*Tel Aviv University and CEPR; fersht@post.tau.ac.il.
**gandal@post.tau.ac.il.

We are grateful to the editor, James Hosek, and three anonymous referees whose detailed comments and suggestions significantly improved the article. We are also grateful to participants at several conference and university seminars for helpful comments. We thank Rafi Aviav and Sagit Bar-Gill for very helpful research assistance. Research grants from Microsoft and the Net Institute (www.netinst.org) are gratefully acknowledged. Any opinions expressed are those of the authors.

The focus in the literature is on spillovers between innovations. In the basic setup, an innovation by one firm "spills over" to other firms that enjoy some of its benefits in terms of lower cost, better knowledge, or some advantage in developing a new technology (e.g., d'Aspermont and Jacquemin, 1988). In a model of such research spillovers confined to "connected" firms, Goyal and Moraga-Gonzalez (2001) examine the interaction between the architecture of the collaboration network in oligopolistic markets and firm incentives to invest in R&D.[1]

Our focus in this article is on spillovers that occur through direct interaction between researchers or developers. Commercial and academic research is typically done by teams. The typical R&D project involves teams of researchers who work together on the same project. Working in teams involves exchanging ideas and sharing information. Whenever coworkers collaborate on a joint R&D project, they create knowledge spillovers. Participants of such research teams carry over their knowledge to other teams and other projects that they are involved with.

Even when it is the developers who facilitate the spillovers, the question is whether these developers "learn" from working on a particular project or whether they learn from other individuals who collaborate with them. In the former case, we will say that there are "project spillovers"; in the latter case, there are "contributor spillovers." There is a fine distinction between these two types of spillovers, and one of the objectives of this article is to highlight the difference between them and to demonstrate that they can be empirically distinguishable.

Regardless of whether there are project spillovers or contributor spillovers, knowledge spillovers can be either direct or indirect. Consider, for example, project spillovers. Direct spillovers occur when projects have a common developer who transfers knowledge from one project to another. That is, a developer takes the knowledge that he acquires while working on one project and implements it on another project. But knowledge may also flow between projects even if they are not connected. The indirect route occurs whenever a developer who learns something from participating in one project takes the knowledge to a second project and "shares" it with another developer on that project, who in turn uses it when he works on a third project. In such a scenario, knowledge flows from the first project to the third project, even though they do not have any developers in common. Clearly, such indirect spillovers may be subject to decay depending on the distance (the number of indirect links) between the projects.

Detailed information about R&D projects is typically hard to obtain. In particular, information regarding the identity of developers who participate in each project is not available. This information is essential in constructing the network of collaboration which is the base of our study of knowledge spillovers. The recent "open-source revolution" provides a unique data set in which we are able to identify the developers who participate in each project.

The open-source model is a form of software development with source code that is typically made available to all interested parties.[2] The open-source model has become quite popular and is often referred to as a movement with an ideology and enthusiastic supporters.[3] At the core of this process is a decentralized production process: open-source software development is done by unpaid software developers.[4] Because there are many such projects, these developers may be involved in more than one project and may work with different groups of codevelopers on various open-source projects.

The article uses data from sourceforge.net to construct a two-mode network of open-source software (OSS) projects and developers. Sourceforge.net is the largest repository of OSS code

[1] See also König et al. (2008) for a model of R&D network formation in which firms are engaged in pairwise R&D collaboration.

[2] Open source is different from "freeware" or "shareware." Such software products are often available free of charge, but the source code is not distributed with the program and the user has no right to modify the program.

[3] See, for example, Raymond (2000) and Stallman (1999).

[4] Having unpaid volunteers is puzzling for economists. Although several authors have written about the possible motivation of OSS contributors (see, for example, Harhoff, Henkel, and von Hippel, 2003; Lakhani, and Wolf, 2005; Lerner, and Tirole, 2002; Hertel, Niedner, and Herrmann, 2002), the puzzle essentially remains unsolved; hence, it is not clear how to construct a structural model of these activities.

and applications available on the Internet, with 114,751 projects and 160,104 contributors (in June 2006). Each Sourceforge project page links to a "developer page" that contains a list of registered team members.[5] As the development of the OSS projects is done in the public domain and the developers can be identified by their email addresses, we can use this information to construct the two-mode network of projects and developers. For the "project network," we say that two OSS projects are connected if there are developers who participate in both projects. In the related "developers network," two developers are connected if they work on the same OSS project.[6] Interestingly, both the project network and the contributor network consist of one "giant" connected component and many smaller unconnected networks.

It is not easy to measure the success of open-source software. Unlike commercial software, open-source software is not sold or licensed, and there are no revenues or any measures of economic success. One way to measure project success is to examine the number of times a project has been downloaded. Although this is not always an ideal measure, downloads are a very good measure of success for open-source software.[7]

The objective of our article is to improve our understanding of knowledge spillovers by examining their role in the development and success of OSS projects. Our focus is on the role of direct and indirect spillovers and the relative importance of project spillovers and contributor spillovers. We show that whenever there are direct project spillovers, there should be a positive correlation between project success and the *degree* of the project; this is intuitive, as the *degree* of a project is the number of projects with which the project has a direct link (common developers). When there are indirect project spillovers as well, we show that there should be a positive correlation between project success and project *closeness centrality*. This is intuitive as well, because *closeness centrality* is the inverse of the sum of all distances between the project and all other projects; thus, it measures how far each project is from all the other projects in the network. (We show in Section 3 that *closeness centrality* captures both direct and indirect spillovers.) In an analogous fashion, direct and indirect contributor spillovers are related to the contributor *degree* and contributor *closeness centrality*.

We first empirically examine the association between project success and network measures. We show that network architecture is indeed associated with project success: projects in the giant component have on average four times more downloads than projects outside the giant component. Further, we find that *closeness* is positively and significantly associated with higher downloads. This result is consistent with the presence of both direct and indirect project spillovers. There is not always, however, a positive association between the *degree* of the project and project success. This result is consistent with no evidence for hyperbolic (i.e., especially strong) direct spillovers.

We then empirically examine the association between contributor spillovers and project success. Interestingly, we find that none of the contributor centrality measures are positively associated with success; therefore, we find no association between contributor knowledge spillovers and project success.

Finally, we change the definition of a link and define projects to be "strongly" linked if and only if they have at least two contributors in common; we obtain a dramatic effect on network structure. In this new network, the largest component of strongly connected projects consists of only 259 projects (versus 27,246 projects in the giant component in the previous network). We show that strong connections matter and that there is a large difference between the average (and the median) number of downloads between projects in the large connected component in the strongly connected network and other projects in the giant component.

[5] Sourceforge.net facilitates collaboration of software developers, designers, and other contributors by providing a free of charge centralized resource for managing projects, communications, and code.

[6] One can actually construct a weighted network where the weight of a link in the project network is the number of developers who jointly participate in two projects and the weight of a link in the contributor network is the number of projects on which two developers work together.

[7] Downloads are also often used as a measure of impact of academic articles on the web. The Social Science Research Network, for example, provides information on the number of downloads for the papers on its website.

Our article is related to Grewal, Lilien, and Mallapragada (2006), who use Sourceforge data and investigate (using a sample of 108 open-source projects) how the network embeddedness of projects and project managers influences the success of projects. Although the paper uses some of the network measures that we employ, the paper addresses very different issues. Our article focuses on knowledge spillovers that occur through the interaction between different researchers or developers who collaborate on different research projects. Our article also distinguishes between project spillovers and contributors' spillovers and between direct and indirect spillovers.

Academic research is another area in which contributors' identities are publicly observed. For example, Goyal, van der Leij, and Moraga-Gonzalez (2006) construct the coauthorship network in economics using data on all published papers that were included in EconLit from 1970 to 2000 and study the properties of this network. Our focus, however, is not on the properties of the network per se, but rather on the relationship between network architecture and success.

Our article is also related to Calvo-Armengol, Patacchini, and Zenou (2009) and Ahuja (2000), who also consider the relationship between network structure and performance. Calvo-Armengol et al. use data on an adolescent friendship network and focus on how the existing network structure affects pupils' school performance. Ahuja (2000) examines the relationship between the network formation of technical collaboration among firms in the chemical industry (from 1981 to 1991) and innovation, as measured by U.S. patents. Other papers also relate to the literature that study the effect of network structure on behavior (e.g., Ballester, Calvo-Armengol, and Zenou, 2006; Calvo-Armengol and Jackson, 2004; Ioannides and Datcher-Loury, 2005; Goeree et al., 2010; Jackson and Yariv, 2007; Karlan et al., 2009).[8]

This article is also related to the large literature identifying neighborhood effects, non-market interactions, and spillovers in public, labor, and education settings. A primary concern in this literature is the endogenous determination of spillovers—good examples are Manski (2000), Sacerdote (2001), and Angrist and Lang (2004). This literature has focused on finding institutional settings that impose a level of randomness that allows one to eliminate the selection effect and identify the causal effect of neighbors on each other. For instance, Sacerdote uses random assignment of roommates for Dartmouth College freshmen. The issue of causality is important in our work as well. This article does not have an analogous source of randomness or exogeneity, but rather uses various cuts of the data to evaluate the importance of endogeneity bias in driving the results. In the industrial organization (IO) literature, Rysman (2004) considers network effects.

2. The two-mode network of contributors and projects

■ We obtained our data by "spidering" the website sourceforge.net, which is the largest OSS development website.[9] The data were retrieved from sourceforge.net during June 2006 and includes 114,751 projects and 160,104 contributors who were listed in these projects.[10] The contributors are identified by unique user names they chose when they registered as members in sourceforge. The site's information structure is rooted in projects. The interface of sourceforge.net allows almost all of the information about the projects to be viewed by anyone.[11] Each project has a project page, which is a standardized home page that links to all the services and information made available by sourceforge.net for that project. The project page itself contains important descriptive information about the project, such as a statement of purpose, the intended audience, license, operating system, and so forth.

Each project page links to a "statistics page" that shows various activity measures, such as the number of downloads. Each project page also links to a "developers page" that has a list of

[8] For more general surveys on the role of social networks in the functioning of the economy, see Jackson (2007, 2008) and Goyal (2007), and for general methods and applications see Wasserman and Faust (1994).

[9] "Spidering" is a term used to describe recursive algorithms used to traverse a website page by page and automatically extract desired information based on forms and content pattern.

[10] We surveyed all the projects and contributors that were registered in sourceforge.net at that date.

[11] A very small number of projects block certain data from being accessed by anyone who is not a project team member.

TABLE 1 The Distribution of Contributors per Project and Projects per Contributor

Project Network		Contributor Network	
Contributors per Project	Number of Projects	Projects per Contributor	Number of Contributors
1	77,571	1	123,562
2	17,576	2	22,690
3–4	11,362	3–4	10,347
5–9	6,136	5–9	3,161
10–19	1,638	10–19	317
20–49	412	20–49	26
≥50	56	≥50	1
Total projects	**114,751**	Total contributors	**160,104**

registered team members. This list is managed by the project administrators, who are also listed as team members. The assumption in this article is that the site members who are listed as project team members were added to the list because they made a contribution to the project that involved investment of time and effort. A project is thus seen as a collaborative effort by its team members, or *contributors*.

The data we obtained from sourceforge.net form a two-mode network of projects and contributors. A two-mode network is a network partitioned into two types of nodes, projects and contributors. We can use the two-mode network to construct two different one-mode networks: (i) the contributor network and (ii) the project network.[12]

Contributor network:

- The nodes of this network are the contributors, that is the distinct names (or emails) of the contributors.
- There is a link between two different contributor nodes if the two contributors participated in at least one OSS project together.
- Each link may have a value which reflects the number of projects in which the contributors jointly contributed.

Project network:

- The nodes of this network are the OSS projects.
- There is a link between two different project nodes if there are contributors who participate in both projects.
- Each link may have a value which reflects the number of contributors who participate in both projects.

Table 1 shows the distribution of contributors per project and projects per contributor for the two-mode network at sourceforge.net.

Observation 1: (i) Most of the OSS projects are not carried out by large teams of contributors. On average, there are 1.4 contributors per project. More than two thirds of the projects have only one contributor[13] and only 1.8% of the projects have 10 or more contributors. (ii) Most of the contributors (90%) participate only in one or two OSS projects.

[12] We construct our project network by defining two projects as linked if there are contributors who work on both of them. One can construct different types of networks based on common application, language, and so forth; that is, two projects are connected if they are written for the same application. In our empirical analysis, we control for these variables. Although defining networks based on application and language does capture some aspects of knowledge spillovers, the thrust of our research is on knowledge spillovers created by individuals. We thus focus on the networks that are defined by having common contributors.

[13] Although these projects do not provide links between contributors, such contributors who work on multiple projects provide links among projects.

TABLE 2 Development Status

Development Status	Relative Frequency in Single-Contributor Projects (%)	Relative Frequency in Multicontributor Projects (%)
1 – Planning	21	21
2 – Pre-alpha	17	16
3 – Alpha	18	17
4 – Beta	22	23
5 – Production/stable	18	20
6 – Mature	1	2
Inactive	2	2

Table 1 tells an interesting story about the world of OSS projects. Given the excitement generated by open-source software, one might imagine a world in which there is an army of contributors who work on many different projects; the reality is different. As Observation 1 indicates, 68% of the projects hosted at sourceforge.net have just a single contributor. An additional 15% of the projects have 2 contributors. Hence, more than 80% of the projects have either 1 or 2 contributors. At the other end of the spectrum, there are 1,638 projects with 10–19 contributors and 468 projects with 20 or more contributors. Table 1 also indicates that 77% of the contributors worked on a single project and more than 90% of them worked on just 1 or 2 projects. There are a small number of devoted contributors who work on many projects: there are 3,505 contributors who work on 5 or more projects and 344 contributors who work on 10 or more projects. This suggests that the world of open-source projects is much less strongly connected than we might have believed.

Because our data are a snapshot taken at a particular date, it is possible that projects with one contributor are projects at an early stage of development. There are six levels of development that range from the planning stage to a mature status. There is an additional status reserved for projects that are inactive. Table 2 provides the distribution of the development status for single-contributor and multicontributor projects.

Observation 2: The distributions of the development status for single-contributor and multicontributor projects are similar. Thus, the possibility that the single-contributor projects are in some way infant projects seems remote.

Of course, in our analysis, we will control for the time for which the project has been in existence and the stage of development, and we will examine whether our results are robust to the exclusion of single-contributor projects.

□ **The network of contributors.** For the contributor network, there is a link between contributors *i* and *j* if they have worked on at least one project in common. The set of contributors can be divided into components such that all of the contributors in a component are connected to one another and there is no sequence of links among contributors in different components. The distribution of the components is shown in Table 3. There is a "giant" component, which consists of 55,087 contributors, or approximately 45% of the contributors, and many small components as well.

For every contributor in the network, we can define the *degree* as the number of links between that contributor and other contributors in the network.[14] Table 4 shows the distribution of *degree* in the contributor network. There are 47,787 contributors who work only in single-contributor projects and therefore have a *degree* of zero. At the other end of the spectrum, 491 contributors worked on projects in common with more than 256 other contributors.

[14] Hence, a contributor who worked on a single project with four other contributors has a degree of four. Similarly, a contributor who worked on two projects, each of which had two additional contributors (who only worked on one of the two projects), would also have a contributor degree equal to four.

TABLE 3 Distribution of Component Size

Component Size (Contributors)	Components (Subnetworks)
55,087	1
196	1
65–128	2
33–64	27
17–32	152
9–16	657
5–8	2,092
3–4	4,810
2	8,287
1	47,787

TABLE 4 Distribution of Degree

Degree	Number of Contributors
0	47,787
1	22,133
2	14,818
3–4	20,271
5–8	20,121
9–16	16,228
17–32	10,004
33–64	5,409
65–128	2,040
129–256	802
257–505	491

TABLE 5 Distribution of Component Size

Size	Connected Components
27,246	1
17–27	36
9–16	234
5–8	1,013
3–4	3,419
2	8,020
1	51,093

Observation 3: Despite the fact that more than 90% of the contributors worked only on one or two projects more than a third of the contributors belong to a giant component of 55,087 connected contributors. The other connected components are relatively small with relatively few contributors.

□ **The network of projects.** In the project network, a node is a project and there is a link between two projects if and only if there are contributors who have contributed to both projects. Table 5 shows the distribution of connected components of the project network. Table 6 shows the distribution of degree for the project network. The degree of a project is the number of other projects with which that project has a link.

Observation 4: (i) The project network has a very special structure. There is one giant connected component with 27,246 projects (approximately 24% of the projects at the Sourceforge website)

TABLE 6 Distribution of Degree

Degree	Number of Projects
0	51,093
1	22,926
2	12,709
3–8	22,004
9–32	5,649
33–64	290
≥65	80

and many very small unconnected components. Remarkably, the second largest connected component has only 27 projects. (ii) Two thirds of the projects have a degree less than or equal to 1. At the other end of the spectrum, 370 projects have a degree greater than 32.

3. Knowledge spillovers

■ Learning is done by individuals. It is possible to distinguish between two types of spillovers: project spillovers and contributor spillovers. In both cases, it is the contributors themselves who facilitate the spillovers. But the question is: do contributors "learn" from working on a particular project, or do they learn from other individuals who collaborate with them? In the former case, we will say that there are "projects spillovers"; in the latter case, there are "contributor spillovers." It is typically very difficult to distinguish between these two types of spillovers. We are able to examine this issue empirically because the unique data set we constructed has detailed information on both projects and contributors.

We start by considering spillovers between projects. Such spillovers can either be direct or indirect. Direct spillovers occur when projects have a common developer who transfers information and knowledge from one project to another. Project spillovers may also be indirect, when knowledge is transferred from one project to another even when the two projects are not directly linked (i.e., they have no common contributor). The indirect spillover route involves a learning mechanism such that a developer who participates in project X acquires knowledge when he participates in project Y and then employs the knowledge on project X. Another project-X developer (who does not work on project Y) then uses that knowledge on project Z. This distinction can be summarized by the following definition.

Definition 1: (i) **Direct project spillovers** exist whenever there are knowledge spillovers between projects that are directly connected, that is they have common contributors. (ii) **Indirect project spillovers** exist whenever there are knowledge spillovers between projects that are not directly connected, that is, projects for which there are no common contributors.

An alternative approach would be to assume that contributors accumulate knowledge and that there are knowledge spillovers among the contributors. This case involves the contributor network. We again can distinguish between direct and indirect knowledge spillovers allowing contributors to learn indirectly from other developers even though they do not work together on the same project.

Definition 2: (i) **Direct contributor spillovers** exist whenever there are knowledge spillovers between contributors who are directly connected, that is they work together on the same project. (ii) **Indirect contributor spillovers** exist whenever there are knowledge spillovers between contributors who are not directly connected.

Because we do not directly observe spillovers, we will examine the relationship between the network structure and project success in order to identify the relative importance of the different types of knowledge spillovers. We briefly discuss the network measures that are relevant for our

analysis. We define these measures in terms of the project network case (the definitions for the contributor network are analogous).

(i) The *degree* of a project is the number of projects with which it has a direct link or common developers.

(ii) *Closeness centrality* is defined for every project as the inverse of the sum of all distances between the project and all other projects multiplied by the number of other projects. Intuitively, *closeness centrality* measures how far each project is from all the other projects in the network. According to this definition, *closeness centrality* lies in the range [0,1]. Formally, for any two nodes $i, j \in N$, the distance or *degree* of separation between them (denoted $d(i, j)$) is the length of the geodesic between them where a geodesic is the shortest path between two nodes. *Closeness centrality* is calculated as[15]

$$C_C(i) \equiv \frac{(N-1)}{\sum_{j \in N} d(i, j)} \tag{1}$$

Analogously, we can construct the contributor network and derive the network characteristics for each contributor, that is, the *degree* and the *closeness centrality* of each contributor.

We now briefly discuss the relationship between the type of spillovers we have in mind and the network characteristics that we use. We discuss these relationships for the project spillover case, but an analogous structure exists for the contributor spillover case. Assume that all the projects are symmetric except for their position in the network and that the expected success level of each project (without any spillovers) is given by α. Assume further that each project also receives a positive (constant) spillover from all connected projects. Thus, the success level of each project i is α plus β multiplied by the number of direct links of each project,

$$\mathrm{S}_i = \alpha + \beta^* \mathrm{D}_\mathrm{i}, \tag{2}$$

where D_i is the *degree* of project i in the network and β is the magnitude of the direct spillovers.

Now assume that the project also enjoys positive spillovers from projects that are indirectly connected, but that these spillovers are subject to decay. We assume that the greater the distance between the project in the project network, the fewer the indirect spillovers. Formally, when the distance between project i and j is denoted as $d(i,j)$, we assume that the expected success of each project is $S_i = \alpha + \sum_j \gamma / d(i, j)$, where γ is the magnitude of the spillovers.[16] Using (1) above, project i's success can be rewritten as

$$S_i = \alpha + \gamma C_c(i)/[N-1], \tag{3}$$

where $C_c(i)$ is the measure of project i's *closeness centrality*. When (3) holds, the spillovers are fully captured by the *closeness* measure of each project. Despite this, there are both direct and indirect spillovers.

It is possible that direct and indirect spillovers have different impacts. We can capture this by the following (more general) specification:

$$S_i = \alpha + \gamma C_c(i)/[N-1] + \beta D_i. \tag{4}$$

When $\beta = 0$, there are no additional spillovers from directly connected projects above and beyond those captured by its *closeness* measure and (4) reduces to (3). When $\beta > 0$, the spillovers have a "hyperbolic" structure: there are additional spillovers from directly connected projects.

We will estimate equation (4) and examine whether, accounting for the effect of all the control variables (which we will add to the regression), *degree* and *closeness* are associated with

[15] See Freeman (1979) and Wasserman and Faust (1994).

[16] This is only one possible type of decay. Clearly, it is possible to assume different types of decay.

a larger number of downloads. If the estimated coefficient on *closeness* is positive and significant, there is evidence for both direct and indirect project spillovers.[17] If the estimated coefficient on *degree* (β) is also positive and significant, there is evidence for a hyperbolic structure with especially strong direct spillovers among connected projects. If $\gamma = 0$ but β is positive, we have evidence for direct spillovers only, that is, there are no indirect spillovers. If both γ and β equal zero, there is no evidence for any direct or indirect spillovers.

4. Direct and indirect project spillovers: empirical analysis

■ We wish to examine whether and what type of knowledge spillovers plays is a role in the development of OSS projects. We start our empirical analysis by defining a measure of success and different control variables that identify the important characteristics of OSS projects. Our analysis of the type of knowledge spillovers will be carried out in the following stages. In this section, we will examine the association between project network measures and success. We will then examine the association between contributor network measures and success (Section 5) and then the importance of thick or strong ties among projects (Section 6).

□ **Measuring success/output in the project network.** Defining or measuring the success of an open-source project is problematic. There are no prices and no sales. The projects are in the public domain and there is no need to request permission or provide payment for using OSS. One way to measure project success is to examine the number of times a project has been downloaded. Although this is not always an ideal measure, downloads are a very good measure of success for open-source software. Unlike downloads of academic papers, users will not typically download a project (and its code) unless it will be useful to them for some task.[18]

Every month, the sourceforge.net staff chooses a "project of the month." Although we do not know the exact criteria that are employed in choosing the project of the month, these projects are likely to be very "successful." We obtained data on the project of the month for the 42 month period ending in June 2006. The project of the month projects have an especially large number of downloads.[19] Project of the month projects are typically in advanced stages (stages 4, 5, and 6); 38 of the 42 projects of the month are either in stage 4, stage 5, or stage 6. The 38 projects of the month in advance stages had on average 6,028,560 downloads, versus 30,206 downloads (on average) for the other 35,821 projects in advanced stages. The median number of downloads for projects of the month in advance stages was 1,154,469 versus 483 for other projects in advance stages. This suggests that the number of project downloads is an attractive measure of use and value.

There are several different download measures that we can use: (i) the total number of downloads since the project was initiated at sourceforge.net, (ii) the maximum number of downloads in any month, and (iii) the number of recent downloads. The correlation among these download measures is, however, quite high. Because it contains the most information, we chose to use the total number of downloads in our analysis. Henceforth, when we refer to downloads, we mean the total number of downloads and denote *downloads* as the total number of downloads for the 42 month period for which we have data. We further define $ldownloads \equiv \ln(1 + downloads)$, where "ln" indicates the natural logarithm.

[17] There are possible alternative explanations for a positive association between *degree* and *closeness* and downloads; we discuss this issue in Section 4.

[18] In some cases, the number of downloads is small relative to the number of contributors. In such cases, the number of downloads may be affected by the fact that developers may need to download the code of the project when working on the project. When we restrict our analysis to projects with more than 200 downloads, and a download/contributor ratio of at least 10/1 (so that the number of downloads is at least an order of magnitude larger than the number of developers), our results remain qualitatively unchanged; hence, our results are robust to the possibility of developer downloads. See Section 4.

[19] Given that there are only 42 such projects of the month, we cannot use this as our measure of success.

□ **Network and control variables (project characteristics)** For our empirical analysis, we employ the project network variables $ldegree = \ln(1 + degree)$ and $lcloseness = \ln(0.05 + closeness)$,[20] where degree and closeness are defined in Section 3. In addition to downloads and the network variables, we have data for a group of control variables that include the amount of time that the project has been in existence, the stage of development, the number of operating systems for which the program was written, the number of languages in which the program is written, as well as several other control variables:

- The variable *years_since* is the number of years that have elapsed since the project first appeared at sourceforge: $lyears\_since = \ln(years\_since)$.
- The variable *cpp* is the number of contributors who participated in the project: $lcpp = \ln(cpp)$.
- The dummy variable *ds_j* refers to the stage where *j* ranges from one to six. There is an additional stage, denoted *inactive*, which means the project is no longer active. See Table 2. A few of the projects are considered to be in multiple stages. Hence, for a particular project, it is possible that both ds_3 and ds_4 could be equal to one.
- The variable *count_trans* is the number of languages in which the project appears including English. Virtually all of the projects (95%) are available in English. The other popular languages include German (5% of the projects), French (4%), and Spanish (3%). $lcount\_trans = \ln(count\_trans)$.
- The variable *count_op_sy* is the number of operating systems (i.e., formats) in which the project is compatible. Some of the projects are available for several operating systems. The main operating systems in which the projects were written include Windows (32% of the projects), Posix (26%), and Linux (21%) $lcount\_op\_sy = \ln(count\_op\_sy)$.
- The variable *count_topics* is the number of topics included in the project description. Popular topics include the Internet (16% of the projects), software development (14%), communications software (11%), and games and entertainment software (10%). $lcount\_topics = \ln(count\_topics)$.
- The variable *count_aud* is the number of main audiences for which the project was intended. The main audiences are developers (35%), end users (30%), and system administrators (13%). Some of the products are intended for multiple main audiences whereas other projects are not intended for these main audiences but rather just for niche audiences, that is, just for a particular industry (i.e., telecommunications) or just for very sophisticated end users. $lcount\_aud = \ln(1 + count\_aud)$.

Clearly, there are different ways to construct these variables. For example, we could have simply counted the key operating systems, or used dummy variables for these operating systems. Similarly, we could have defined dummy variables for main audiences or we could have added up the number of main audiences together with the number of niche audiences. We chose the definitions that seemed most natural. Our main results regarding the number of contributors and the network variables are robust to alternative definitions of these control variables.[21]

□ **Empirical results.** We estimate a simple log/log model of the form $ldownloads_i = \alpha + \beta N_i + \gamma C_i + \varepsilon_i$, where the subscript i refers to the project. N_i is the natural logarithm of the network variables and C_i is the natural logarithm of the control variables.[22] For binary ($\{0,1\}$) variables, we of course do not employ logarithms; ε_i is a random error term.

[20] The reason we add such a small number is that the mean value of *closeness* is 0.14.

[21] Contributor effort is not observable. As we discussed in the Introduction, the main reward to OSS contributors is being included in the list of contributors. Thus, the incentive they have is to provide the sufficient effort to accomplish this status. Hence, effort is not likely correlated with network measures or the control variables—and hence, the absence of data on effort does not bias our results.

[22] The relationship between the number of contributors and downloads is likely nonlinear: additional contributors are likely associated with a larger number of downloads, but the marginal effect of each additional contributor declines as the number of contributors increases. The same is likely true for the relationship between the network variables and

We have data on 114,450 observations for all of the network variables as well as on *years_since*.[23] However, data on the stage of development and the count variables are incomplete; data on all of the control variables are available only for 66,511 projects. Because there is no selection issue,[24] we use only the data on the 66,511 projects for which we have complete information. We use information from all the projects to construct the network variables that are included in the database. We first conduct an analysis using these projects and examine the association between *degree* (and the control variables) and success. We then examine the giant component in detail (18,697 projects for which there is complete information), which enables us to include *closeness* in the analysis.[25]

The effect of *degree* (as well as other effects) may depend on whether the project is in the giant component or not; we therefore introduce the variable giant_comp, which is a dummy variable that takes on the value one if the project is in the giant component, and the value zero otherwise. In order to allow for the possibility that the association between *degree* and downloads and between the number of contributors and downloads depends on whether the project is inside or outside of the giant component, we also include the following interaction variables in the analysis:

- *lgiant_degree* = ldegree ∗ giant_comp,
- *lgiant_cpp* = lcpp ∗ giant_comp.

By including the interaction variables, we allow for the possibility that there will be different download "elasticities" for projects inside and projects outside of the giant component.[26]

Descriptive statistics of the variables are shown in Table A1 in the Appendix. Table A1 shows that projects in the giant component have on average many more downloads than projects outside of the giant component (42,751 vs. 10,959). Further, projects in the giant component are on average (i) older than projects outside of the giant component (3.63 years vs. 2.70 years), (ii) have more contributors (3.84 vs. 1.61), and (iii) have a larger *degree* (6.26 vs. 1.18).[27] The results of a regression with all 66,511 observations are shown in the first column of Table 7.

The effect of the number of contributors. The estimated coefficients show that the association between downloads and the number of contributors is positive—projects with more contributors have a greater number of downloads. For projects outside of the giant component, the estimated "contributor" elasticity is 0.46. This effect is statistically significant. The estimated contributor elasticity is virtually twice as large for projects in the giant component: 0.90 (0.46 + 0.44). The difference in the estimated contributor elasticity between projects in the giant component and projects outside of the giant component is statistically significant: additional contributors are associated with greater increases in output for projects in the giant component than in the non-connected component. This result obtains despite the fact that there are many more contributors (on average) for projects in the giant component (3.84 vs. 1.61).

The effect of the project's degree. The association between the *degree* of the project and the number of downloads is positive and statistically significant both for projects inside the giant component and for projects outside of the giant component. For projects outside of the giant

downloads as well. This suggests that a "log/log" model is appropriate. We examine alternative functional forms below. There, we indeed find that the log/log specification has a higher adjusted R^2 that, both the log/linear and linear/linear specifications.

[23] There are 114,751 total projects, but we are missing data on downloads for a small number of them (301).

[24] See Griliches (1986) and Greene (1993).

[25] The values of *degree* and *closeness centrality* are calculated using the software program Pajek, which is a software program for large-network analysis. See pajek.imfm.si/doku.php.

[26] The addition of different slopes for the control variables based on whether the project was inside or outside of the giant component has no effect on the main results regarding the number of contributors and the *degree* of the project.

[27] Correlations among the network centrality variables in the giant component are shown in Table A2.

TABLE 7 Regression Results: Dependent Variable: *ldownloads*

Dependent Variable: *ldownloads*	Regression 1 (All 66,511 Projects)		Regression 2 (Giant Component: 18.697 Projects)	
Independent variables	Coefficient	T-Statistic	Coefficient	T-Statistic
Constant	0.72	17.76	1.45	3.62
lyears_since	1.42	60.66	1.68	31.08
lcount_topics	0.23	9.07	0.18	3.59
lcount_trans	0.35	11.73	0.45	8.15
lcount_aud	0.36	10.44	0.44	5.85
lcount_op_sy	0.11	5.95	0.18	5.00
ds_1	−1.96	−60.57	−2.01	−31.90
ds_2	−0.60	−17.58	−0.78	−11.50
ds_3	0.89	25.83	0.66	9.95
ds_4	1.86	57.21	1.80	29.27
ds_5	2.72	79.97	2.61	40.96
ds_6	2.12	27.07	2.03	15.35
inactive	0.45	6.11	0.39	2.75
lcpp	0.46	18.71	0.87	29.34
ldegree	0.19	9.45	0.079	2.10
giant_comp	−0.21	−3.86		
lgiant_cpp	0.44	12.05		
lgiant_degree	−0.05	−1.26		
lcloseness			0.69	3.21
Number of observations	66,511		18,697	
Adjusted R^2	0.41		0.40	

component, the *degree* elasticity is 0.19, whereas the *degree* elasticity for projects inside the giant component is 0.14. Both of these magnitudes are statistically significant from zero; the difference in the magnitudes is not significantly different from zero.

The effect of the control variables. The estimated coefficient of *lyears_since* is positive (1.42) and statistically significant. Projects that have been active longer have more downloads, and the estimated coefficient suggests that a doubling of the time a project has been active is associated with 142% more downloads. The estimated coefficients on the stage variables have the expected signs. By and large, projects that are in more advanced stages are associated with more downloads. Similarly, projects written for several operating systems, projects available in more languages, projects written for more main audiences, and projects that span more topics are associated with more downloads as well.

Observation 5: (i) Projects in the giant component have on average (four times) more downloads than projects outside of the giant component. (ii) Projects with more contributors have a greater number of downloads, and this effect is stronger in the giant component. (iii) The association between the degree of the project and the number of downloads is positive and statistically significant (both inside and outside the giant component).

From Observation 5, we can conclude that the network architecture does affect the number of downloads, which suggests that there are knowledge spillovers among the projects.

Our next step is to introduce the variable *lcloseness* into the regression (see the second regression in Table 7). Because *closeness* is only comparable across linked networks, this regression is done for the giant component only (18,697 observations). Note that in the new regression the estimated contributor elasticity (0.87, t = 16.71) and the estimated coefficient on *lyears_since* (1.68, t = 31.08) are again positive and statistically significant. The estimated coefficients on the stage and count variables again have the expected signs and are qualitatively similar to those in the first regression in Table 7.

This regression also shows that the estimated *closeness* elasticity (0.69, t = 3.21) is statistically significant. Controlling for *closeness*, there is still a positive association between the number of downloads and the degree of the project. The estimated *degree* elasticity (0.079, t = 2.10) is also statistically significant in this regression.

Observation 6: *Closeness* is positively and significantly associated with higher downloads. This suggests that indirect spillovers are important.

The second regression in Table 7 indicates that the estimated coefficient on *degree* is also positive and significant. This suggests that there are hyperbolic direct spillovers as well. We now will conduct several robustness tests in order to examine whether these results are robust.

□ **Robustness analysis.** In this section, we will examine whether the results in the second regression in Table 7 are robust by examining established projects only, projects with more than one contributor, and projects with a relatively large number of downloads. We then examine the robustness of the results to functional form, to possible endogeneities, and to including an additional network centrality measure. We conclude this section by examining alternative interpretations of the results.

Established projects, more than one contributor, and a large number of downloads. Nascent projects may not have reached a steady-state number of contributors. Personnel additions are probably more likely for relatively new products. Here we examine whether our results are robust to using only established projects in the analysis. We also restrict the anaslysis to projects in existence for at least 2 years. For similar reasons, we also restrict the analysis here to projects with more than one contributor. (We focus henceforth on the second regression in Table 7 because this regression includes *degree* and *closeness*.)

In some cases, the number of downloads is small relative to the number of contributors. In such cases, the number of downloads may be affected by the fact that developers may need to download the code of the project when working on the project. Hence, we also restrict our analysis to projects with more than 200 downloads and a download/contributor ratio of at least 10/1 (so that the number of downloads is at least an order of magnitude larger than the number of developers).

When we include all of the above robustness "restrictions" together (projects with more than one contributor, projects in existence for more than 2 years, projects with more than 200 downloads, and a download/contributor ratio of at least 10/1), we are left with 6,397 observations. We again find that the estimated contributor elasticity (0.76, t = 22.21), the estimated *closeness* elasticity (0.71, t = 3.28), and the estimated *degree* elasticity (0.19, t = 5.07) are positive and statistically significant. The robustness analysis thus suggests that the results regarding the contributor elasticity, *closeness*, and *degree* are robust to all of these changes. These results are shown in the first regression in Table A3.[28]

Robustness to functional form. When we run a log/linear regression (the dependent variable remains in logarithms, but the independent variables are in levels), we have the following results: the estimated coefficient on *closeness* is positive and statistically significant both for a regression with all observations in the giant component as well as for a regression (6,397 observations) with all three robustness restrictions discussed above. The estimated coefficient on *degree* is insignificant in a regression with all observations in the giant component; it is positive and statistically significant in a regression with all three robustness restrictions in Table A3.

When we run a linear/linear regression (both the dependent variable and the independent variables are in levels), rather than a log/log regression, the estimated coefficient on *closeness*

[28] The same qualitative results are obtained when we examine these three robust restrictions separately. The estimated contributor elasticity remains positive and statistically significant in all specifications. Because our focus is on degree and closeness, we do not discuss the estimated contributor elasticity in the analysis that follows.

is still positive and statistically significant. The estimated coefficient on *degree* is positive, but no longer statistically significant. This result holds both for a regression with all observations in the giant component (18,697) as well as for a regression (6,397 observations) with all three robustness restrictions.[29]

We can thus conclude that the positive and statistically significant results regarding *closeness* are robust to functional form. The results on *degree* are not completely robust to functional form.

Potential endogeneities. *Degree* could be endogenous in our data set. Here, the interpretation would be that developers may want to be associated with more successful projects. This would make *degree* endogenous. (This is sometimes referred to as the chicken vs. egg issue.)

Closeness could also be endogenous under the following scenario: developers may want to work on a particular project so that a developer on that project can "introduce" them to a developer (on another project) who they would like to meet. Because our network is a fairly thin one (many projects and relatively few developers) and the average project in our data set has fewer than four contributors to a project in the giant component, it is unlikely that this indirect contact mechanism would play any role. It would likely be much easier and much more effective to simply contact the programmer directly. Nevertheless, we wish to address this potential endogeneity as well.

With the exception of Calvo-Armengol et al. (2009), we are not aware of any empirical papers in the social network literature that estimate a structural model and (hence) are able to econometrically deal with the endogeneity issue by using instruments. Unfortunately, neither Calvo-Armengol et al. (2009) nor other theoretical models (such as König et al., 2008) are appropriate for our setting. Even if we could develop a structural model, it would likely depend on variables such as effort or marginal cost that are not observable.

Hence, we must address the potential endogeneity of *degree* and *closeness* in another way. One way to address this issue is indeed to only consider relatively young projects. The "joining popular projects" effect is likely to be less of a factor for relatively young projects. When we run a regression with projects less than 3.63 years old (the mean age of the projects in the giant component) with more than 200 downloads and a download/contributor ratio greater than 10, we find the following[30]: the estimated coefficient on *closeness* remains positive and statistically significant (0.54, $t = 2.37$), while the estimated coefficient on *degree* (0.0038, $t = 0.09$) is not statistically significant. (These results appear in the second regression in Table A3.)

This suggests that *degree* is indeed potentially endogenous. Nevertheless, the estimated coefficient on *closeness* is virtually unchanged. These results suggest that *closeness* is not endogenous and that, despite the potential endogeneity of *degree*, the results for *closeness* remain qualitatively unchanged. Our main conclusion, that there are both direct and indirect spillovers, holds despite the potential endogeneity of *degree*.

The flow of information: betweenness centrality. In this section, we consider another centrality measure—betweenness centrality—and we examine whether our results are robust to its inclusion. Before we define betweenness centrality, we will illustrate this measure by using the (thick) project network shown in Figure A1. We can see that this network has an interesting structure. There are three clusters or groups of highly connected projects.[31] The three clusters remain connected as part of one component only because project 81 is connected to all these three groups. Project 81 has a relatively small *degree*, but its position in the network is unique and central. This position is relevant for an additional type of knowledge spillover. Assume, for example, that the three groups in Figure A1 describe a friendship network among people. Moreover, assume that each cluster in this network is a group of friends who are similar in their backgrounds and preferences. Suppose that the knowledge transmitted in this network is about the quality of a restaurant or a movie. In this case, the information received from members of the same group would be more

[29] The linear/linear specification has a very low adjusted R^2 (0.04). In contrast, the log/linear specification has an adjusted R^2 of 0.25; the log/log specification (regression 1 in Table A3) has an adjusted R^2 of 0.28.

[30] We did include projects with a single contributor here because they are important when examining young projects.

[31] Each has some periphery networks that are connected only to one particular group.

valuable than information received from members of other groups. On the other hand, there are research settings where ideas come from groups of researchers who think and solve problems in different ways. It is possible that in such an environment the more valuable knowledge spillovers come from outside of the research group's inner core. In these cases, the position of project 81 (Figure A1), which is linked to several different clusters of projects, may benefit from valuable knowledge spillovers from the different clusters of projects.

We capture this effect by introducing *betweenness centrality* into our empirical analysis. *Betweenness centrality* is defined as the proportion of all geodesics between pairs of other nodes that include this node.[32] *Betweenness* captures the notion that a node is considered "central" if it serves as a valuable juncture between other nodes. Project 81 in Figure A1 indeed has relatively high *betweenness*. Formally, the *betweenness* of a node i is given by

$$C_B(i) \equiv \frac{\sum_{\substack{j<k \\ i \notin \{j,k\} \subseteq N}} [\gamma_{jk}(i)/\gamma_{jk}]}{(N-1)(N-2)/2}, \tag{5}$$

where γ_{jk} is the number of distinct geodesics between the nodes j and k which are distinct from i, and $\gamma_{jk}(i)$ is the number of such geodesics which include i.[33] When we add *betweenness* to the analysis and run a regression (6,397 observations) with all three robustness restrictions, we find that the estimated coefficient on *degree* is insignificant, while the estimated coefficient on *closeness* remains positive and statistically significant (coeff = 0.45, t = 2.08.). This again suggests that our results on *closeness* are again robust. The estimated coefficient on *betweenness* is positive and statistically significant, suggesting the possibility of an additional type of knowledge spillover. (These results appear in the third regression in Table A3).

Robustness results above show that the estimated coefficient on *closeness* remains positive and statistically significant in all robustness specifications, whereas *degree* becomes insignificant in several instances. Note that a positive coefficient on *closeness* provides evidence for both direct and indirect spillovers. Because the coefficient on *degree* becomes insignificant in several robustness regressions, we do not find convincing evidence for hyperbolic direct spillovers.

Observation 7: Both direct and indirect spillovers are important. *Closeness* is positively and significantly associated with higher downloads. However, we do not find convincing evidence for hyperbolic direct spillovers as, controlling for *closeness*, there is not always a positive association between the degree of the project and the number of downloads.

Alternative interpretations of the results. Positive correlations are, of course, not sufficient for identifying a knowledge spillover. Indeed, the interpretation of a direct knowledge spillover would be problematic if there were only a few highly productive developers and these productive developers signed up for many projects and also caused their projects to have high downloads. In such a case, *degree* would be significant in the regression, yet there would be no knowledge spillover.

We went back and excluded projects that had developers who worked on five or more projects (i.e., "star" contributors). In this new robustness regression, we included the robustness restrictions from above (more than one contributor, projects that were at least 2 years old, projects with more than 200 downloads, and a download/contributor ratio greater than 10.) We had 2,917 observations in this regression. The summary of the regression results (for the network variables) is as follows: both the estimated coefficient on *closeness* (0.77, t = 2.51) and the estimated coefficient on *degree* (0.38, t = 4.54) remain positive and statistically significant.

There is also an alternative explanation (i.e., nonspillover story) regarding the positive correlation between *closeness* and success: if highly productive developers work together (a few

[32] See Freeman (1979) and Wasserman and Faust (1994).

[33] The denominator of (1) is the maximum possible value for the numerator, and thus standardizes the measure in the range [0, 1].

to a project), their projects will be high in "connectedness" because they will be linked to other projects characterized by many links even if there is no spillover. Although this story is plausible in a small, relatively tightly connected network, it is unlikely in our network, which is huge and fairly thinly connected (see Table 1). This suggests that the interpretation of *degree* and *closeness* as knowledge spillovers is reasonable in our case.

5. Contributor network characteristics and project success

■ Up until this point, we have focused on project network characteristics and the way they are associated with the success of projects. Our next step is to focus on contributor network characteristics and to examine their relation to project success.

□ **The effect of contributor characteristics.** We construct the contributor network and derive the network characteristics for each contributor. In order to examine the relationship between these characteristics and project success, we need to look at the group of contributors who participate in each project and define measures that capture the network characteristics of these contributors. For each project, we form a list of contributors and construct the following variables[34]:

(i) the average *degree* of the contributors in a project;
(ii) the average *closeness centrality* of the contributors to a project.

The above variables differ respectively from the *degree* of a project and the *closeness centrality* of a project. For example, consider project A with two contributors (denoted I and II), each of whom works on one other project. This means that project A has a (project) *degree* equal to two. Further suppose that contributor I also works on project B, and that there are three other distinct contributors to project B. Similarly, suppose that contributor II also works on project C, and that there are again three additional distinct contributors to project C. The contributor *degree* of contributor I equals four (because he/she participates with four other contributors in two different open-source projects). Similarly, the contributor *degree* of II is four as well. Hence, the average contributor *degree* of project A is four.

Whereas the *degree* of the project and the average *degree* of the contributors to a project are relatively highly correlated in our data set (0.44),[35] there is virtually no correlation between the *closeness centrality* of a project and the average *closeness centrality* of its contributors (0.03).

We first ran a regression similar to the second regression in Table 7 with the two contributor network variables instead of the two project network variables. Neither the average *closeness centrality* of the contributors to a project (coefficient $= 0.12$, $t = 1.59$) nor the average *degree* of the contributors on a project (-0.019, $t = -0.72$) are statistically significant. When we include both the project and contributor centrality variables in the regression, we find that project centrality measures (*degree* and *closeness*) are again highly associated with success, whereas the contributor centrality variables are not associated with project success.

Observation 8: Our analysis indicates that with respect to OSS development, it is the project spillovers which are important and not the contributor spillovers. Specifically, direct and indirect project spillovers are associated with project success whereas knowledge spillovers between individual contributors are not associated with project success.

Our analysis examined two types of spillovers: spillovers between projects and spillovers between individuals. In both cases it is the contributors themselves who facilitate the spillovers. But the question is: do they learn from working on a particular project or do they learn from other individuals who collaborate with them? Observation 8 states an interesting result: in the world of OSS projects, spillovers occur between projects and not between participants.

[34] Our results are robust to employing the maximum *degree* and maximum *closeness centrality* of the contributors on a project rather than the average *degree* and *closeness centrality* of the contributors on a project.

[35] This is the correlation between the natural logarithm of the variables, because we use those in the analysis.

□ **The star effect.** Some of the contributors in our data set work on many projects. The question is whether these individuals have special talents or abilities that make a significant contribution to the projects in which they participate. We define a "star" as a contributor who worked on five or more projects. Clearly, having a star contributor gives a project more connections with other projects. Indeed, we find that stars are much more common in projects that are in the giant component. Specifically, 45% of the projects in the giant component have at least one star, whereas only 8% of the projects outside of the giant component have a star. An interesting question is whether having a star in the team of developers has an effect on the success of a project.

To examine this, we add a dummy variable (denoted star)—which takes on the value one if the project has at least one star and takes on the value zero otherwise—to the second regression in Table 7. The estimated star contributor is negative (-0.14, $t = -1.94$). Because *degree* and star are highly correlated, one possibility is that any positive effect associated with a star is picked up by *degree*.[36]

Observation 9: After controlling for network centrality measures, star programmers do not make any significant contribution to project success. That is, star programmers do not make a difference beyond that captured by changes in the network measures of the projects in which they participate.

Note that our above conclusion is with respect to having a star working for the project and holding the *degree* and the structure of the network fixed. Clearly, when a star contributor chooses to work for a certain project he changes the network structure and in particular the network characteristics of the project for which he works. Hence, the above observation suggests that any additional contributions made by star contributors are fully taken into account by the change in the network measures.

6. The importance of strong ties

■ So far, we have defined two projects to be linked if there was at least one contributor in common between them. But the potential of spillovers between projects may depend also on the number of contributors who participated in both projects. To capture this effect, we change the definition of a link and focus only on "strong" (or thick) links. Two projects are strongly linked if they have at least two contributors in common. That is, we define a new network in which the nodes are still projects, but the links are only strong (thick) links.

Redefining the network has a dramatic effect on its structure. Previously, in a network in which one contributor in common was sufficient for a link, there was a giant component of 27,246 projects. In the strongly connected network, the largest component of strongly connected projects consists of only 259 projects. There are four other smaller strongly connected components with between 50 and 75 projects. No other components have more than 27 projects.[37]

A comparison between projects in the (i) large strongly connected component, (ii) the four smaller strongly connected components, and (iii) other projects in the giant component is presented below in Table 8.[38]

Observation 10: Strong ties matter. There is a large difference in the median (and the average) number of downloads between projects in the largest strongly connected component, projects in the four smaller strongly connected components, and other projects in the giant component.

[36] The estimated coefficient on *closeness* (0.68, $t = 3.17$) is unaffected by the addition of star.

[37] Figure A1 shows the structure of the largest component in the "strongly connected" network.

[38] The same qualitative result obtains if we restrict the analysis to projects in stages 4–6. In such a case, the medium (mean) numbers of downloads are 11,230 (155,428) for projects in the largest strongly connected component, 2,896 (85,204) for projects in the four smaller strongly connected components, and 1,419 (73,532) for other projects in the giant component.

TABLE 8 Strongly Connected Components versus Other Projects in the Giant Component

Group	Number of Projects	Mean Number of Downloads	Median Number of Downloads	*degree*	Cpp
Strongly connected component[a]	169	120,241	6,491	28.10	13.85
Four smaller strongly connected components	170	61,802	796	22.82	13.62
Other projects in the giant component	18,358	41,859	318	6.07	3.75

[a]We have full data for 169 projects in the largest strongly connected component, and full data for 170 projects in the four smaller strongly connected components.

7. Conclusion

■ Knowledge spillovers are an important part of any learning or R&D process. There are two possible mechanisms that facilitate such spillovers. One possibility is that an individual (or a firm) observes the outcome of an R&D effort of another individual, that is, a new technology or a patent, and learns about its own R&D process. A more direct mechanism is the interaction between different individuals who communicate with their colleagues, exchange emails, switch jobs and projects, and collaborate in different research ventures. The first type of spillover is easier to model as a dynamic process in which any advance or success involving one project positively affects the success of related projects. The second type of learning spillover crucially depends on the "collaboration" network of interaction among individuals who are involved in the learning process. The OSS project network provided a unique opportunity for examining the effect of the properties of both the project network and the contributor network on the success of different projects. We found that there is a positive association between project *closeness centrality* and project success, which suggests the existence of both direct and indirect project knowledge spillovers. We found no evidence, however, for any association between contributor *closeness centrality* and project success, suggesting that contributor spillovers play a lesser role in project success. In light of these results, an important additional step would be to open the "black box" of OSS projects, attempting to collect data on actual communication among team members and (controlling for the structure of the collaboration network) investigate the relationship between actual communication among team members and the success of different projects.

Appendix

FIGURE A1

PROJECTS IN STRONGLY CONNECTED COMPONENT

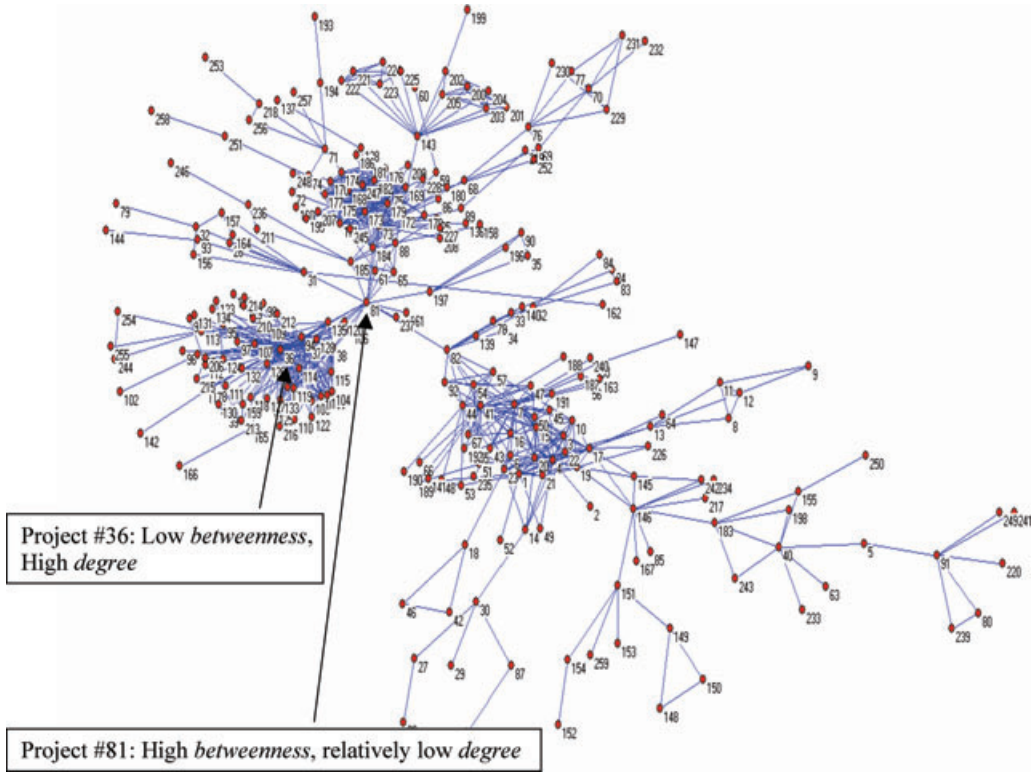

TABLE A1 Descriptive Statistics for 66,511 Projects with Data for All Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
Projects Not in the Giant Component (N = 47,814)				
Downloads	10,959	938,658	0	2.00e+08
years_since	2.70	1.67	0	6.64
count_topics	1.51	0.81	1	7
count_aud	1.21	0.69	0	3
count_op_sy	2.08	1.58	1	21
count_trans	1.27	0.92	1	40
ds_1	0.25	0.43	0	1
ds_2	0.20	0.40	0	1
ds_3	0.20	0.40	0	1
ds_4	0.26	0.44	0	1
ds_5	0.21	0.41	0	1
ds_6	0.02	0.12	0	1
Inactive	0.02	0.14	0	1
Cpp	1.61	1.52	1	42
Degree	1.18	2.14	0	23
Star	0.08	0.28	0	1

(*Continued*)

TABLE A1 Continued

Variable	Mean	Standard Deviation	Minimum	Maximum
Projects in the Giant Component (N = 18,697)				
Downloads	42,751	1,062,802	0	1.18e+08
years_since	3.63	1.70	0.08	6.65
count_topics	1.65	0.89	1	7
count_aud	1.34	0.70	0	3
count_op_sy	2.25	1.69	1	22
count_trans	1.38	1.66	1	45
ds_1	0.22	0.42	0	1
ds_2	0.17	0.38	0	1
ds_3	0.21	0.41	0	1
ds_4	0.30	0.46	0	1
ds_5	0.29	0.45	0	1
ds_6	0.03	0.17	0	1
Inactive	0.03	0.16	0	1
Cpp	3.84	6.72	1	338
Degree	6.26	8.53	1	299
Betweenness	0.00028	0.0015	0	0.12
Closeness	0.14	0.021	0.061	0.22
Star	0.45	0.49	0	1

TABLE A2 Correlation among All Centrality Variables (Giant Component: N = 18,697)

	lcpp	degree	lbetween	lcloseness	star
lcpp	1.00				
ldegree	0.49	1.00			
lbetween	0.71	0.64	1.00		
lcloseness	0.26	0.41	0.36	1.00	
star	0.17	0.74	0.26	0.27	1.00

TABLE A3 Robustness Regressions

Dept Variable: Ldownloads	Robustness Regression 1		Robustness Regression 2		Robustness Regression 3	
Independent variables	Coefficient	T-Statistic	Coefficient	T-Statistic	Coefficient	T-Statistic
Constant	5.75	13.35	6.21	14.55	8.51	16.29
lyears_since	1.08	11.18	0.91	11.40	1.06	10.97
lcount_topics	−0.06	−1.31	−0.06	−1.12	−0.06	−1.23
lcount_trans	0.42	9.61	0.44	8.83	0.41	9.39
lcount_aud	0.21	2.65	0.46	5.62	0.18	2.28
lcount_op_sy	0.26	7.91	0.26	6.62	0.26	7.94
ds_1	−0.46	−6.83	−0.75	−6.23	−0.46	−6.88
ds_2	−0.57	−7.31	−0.52	−4.80	−0.57	−7.68
ds_3	−0.27	−4.35	−0.23	−2.72	−0.28	−4.44
ds_4	0.19	3.45	0.18	2.41	0.18	3.24
ds_5	0.75	12.99	0.54	6.92	0.73	12.80
ds_6	0.73	7.00	0.45	3.06	0.72	6.89
Inactive	−0.018	−0.12	0.02	0.11	−0.041	−0.28
lcpp	0.76	22.21	0.63	19.30	0.59	15.38
ldegree	0.19	5.07	0.0038	0.09	−0.019	−0.43
lcloseness	0.71	3.28	0.54	2.37	0.45	2.08
lbetweenness					0.30	9.21
Number of observations	6,397		4,086		6,397	
Adjusted R^2	0.28		0.25		0.29	

References

Ahuja, G. "Collaboration Networks, Structural Holes, and Innovation: A Longitudinal Study." *Administrative Science Quarterly,* Vol. 45 (2000), pp. 425–455.

Angrist, J. and Lang, K. "Does School Integration Generate Peer Effects? Evidence from Boston's Metco Program." *American Economic Review,* Vol. 94 (2004), pp. 1613–1634.

Ballester, A., Calvo-Armengol, A., and Zenou, Y. "Who's Who in Networks. Wanted: The Key Player." *Econometrica,* Vol. 74 (2006), pp. 1403–1417.

Calvo-Armengol, A., and Jackson, M. "The Effect of Social Networks on Employment and Inequality." *American Economic Review,* Vol. 94 (2004), pp. 426–454.

——, Patacchini, E., and Zenou, Y. "Peer Effect and Social Networks in Education." *Review of Economic Studies,* Vol. 76 (2009), pp. 1239–1267.

d'Aspermont, C., and Jacquemin, A. "Cooperative and Noncooperative R&D in Duopoly with Spillovers." *American Economic Review,* Vol. 78 (1988), pp. 1133–1137.

Freeman, L. "Centrality in Social Networks: Conceptual Clarification." *Social Networks,* Vol. 1 (1979), pp. 215–239.

Goeree, J.K., McConnell, M.A., Mitchell, T., Tromp, T., and Yariv, L. "The 1/d Law of Giving." *American Economic Journal: Microeconomics,* Vol. 2 (2010), pp. 183–203.

Goyal, S. *Connections: An Introduction to the Economics of Networks*. Princeton, NJ: Princeton University Press, 2007.

—— and Moraga-Gonzalez, J.L. "R&D Networks." *RAND Journal of Economics,* Vol. 32 (2001), pp. 686–707.

——, Van Der Leij, M.J., and Moraga-Gonzalez, J.L. "Economics: Emerging Small World." *Journal of Political Economy,* Vol. 114 (2006), pp. 403–412.

Greene, W. *Econometric Analysis,* 2d ed. New York: Macmillan, 1993.

Grewal, R., Lilien, G., and Mallapragada, G. "Location, Location, Location: How Network Embeddedness Affects Project Success in Open-Source Systems." *Management Science,* Vol. 52 (2006), pp. 1043–1056.

Griliches, Z. "Economic Data Issues." In Z. Griliches and M. Intriligator, eds., *Handbook of Econometrics,* Vol. 3. Amsterdam: North Holland, 1986.

Harhoff, D., Henkel, J., and von Hippel, E. "Profiting from Voluntary Spillovers: How Users Benefit by Freely Revealing Their Innovations." *Research Policy,* Vol. 32 (2003), pp. 1753–1769.

Hertel, G., Niedner, S., and Herrmann, S. "Motivation of Software Developers in Open-Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel." *Research Policy,* Vol. 32 (2003), pp. 1159–1177.

Ioannides, Y.M., and Datcher-Loury, L. "Job Information Networks, Neighborhood Effect and Inequality." *Journal of Economic Literature,* Vol. 42 (2005), pp. 1056–1093.

Jackson, M. "The Study of Social Networks in Economics." In *The Missing Links: Formation and Decay of Economic Networks*, edited by J.E. Rauch. New York: Russell Sage Foundation, 2007.

Karlan, D., Mobius, M., Rosenblat, T., and Szeidl, A. "Trust and Social Collateral." *Quarterly Journal of Economics,* Vol. 124 (2009), pp. 1307–1361.

——. "Social Networks in Economics." In Benhabib, Bisin, and M.O. Jackson, eds., *Handbook of Social Economics*. Elsevier, 2008.

—— and Yariv, L. "The Diffusion of Behavior and Equilibrium Structure on Social Networks." *American Economic Review (Papers and Proceedings),* Vol. 97 (2007), pp. 92–98.

König, M., Battiston, S., Napoletano, M., and Schweitzer, F. "The Efficiency and Evolution of R&D Networks." Working Paper no. 08/95, CER-ETH, 2008.

Lakhani, K., and Wolf, R. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free Open-Source Projects." In Feller J. Fitzgerland, S. Hissam, and K. Lakhani, eds., *Perspectives on Free and Open-Source Software*. Cambridge, Mass.: MIT Press, 2005.

Lerner, J., and Tirole, J. "Some Simple Economics of Open-Source." *Journal of Industrial Economics,* Vol. 52 (2002), pp. 197–234.

Manski, C. "Economic Analysis of Social Interactions." *Journal of Economic Perspectives,* Vol. 14 (2000), pp. 115–136.

Raymond, E. "The Cathedral and the Bazaar." (2000). Available at http://www.catb.org~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Rysman, M. "Competition between Networks: A Study of the Market for Yellow Pages." *Review of Economic Studies,* Vol. 71 (2004), pp. 483–512.

Sacerdote, B. "Peer Effects with Random Assignment: Results for Dartmouth Roommates." *Quarterly Journal of Economics,* Vol. 116 (2001), pp. 681–704.

Stallman, R. "The GNU Operating System and the Free Software Movement." In C. Dibona, S. Ockman, and M. Stone, eds., *Open Sources: Voices from the Open Source Movement*. Sepastopol, Calif.: O'Reilly, 1999.

Wasserman, S., and Faust, K. *Social Network Analysis: Methods and Applications,* 2d ed. New York and Cambridge, UK: Cambridge University Press, 1994.