informationresearch

VOL. 21 NO. 1, MARCH, 2016

Academic software downloads from Google Code: useful usage indicators?

Mike Thelwall and Kayvan Kousha

Introduction. *Computer scientists and other researchers often make their programs freely available online. If this software makes a valuable contribution inside or outside of academia then its creators may want to demonstrate this with a suitable indicator, such as download counts.*
Methods. *Download counts, citation counts, labels and licenses were extracted for programs that were both hosted in the Google Code software repository and cited in Scopus.*
Analysis. *Download counts were correlated with Web of Science citations, the distributions of both were compared and common software labels and licencing arrangements were identified.*
Results. *Although downloads correlate positively and significantly with Scopus citations, the correlation is weak (0.3) because some software has a large natural audience outside of academia. There is disagreement on the best licence to use for shared software, with no licence chosen by more than about a fifth of the projects. The most common language label was Java (20%) and, excluding generic computing terms, the most common topic labels were Google (5%), security (3%) and bioinformatics (3%).*
Conclusions. *Download counts can give evidence of wider non-academic uses of software. However, software that is apparently not primarily designed for research but that is nevertheless cited by academics can also attract many downloads. Overall, download counts can be used as an indicator of academic value, but only if contextualised with the purpose of the program.*

change font

Introduction

Academic research is often published in journal articles, monographs, book chapters, and conference papers but the ultimate goal is to produce new and useful knowledge and this can appear in other forms. This diversity is recognised in the UK Research Excellence Framework 2014, for example, with journal articles, books, book chapters, reports, physical artefacts, exhibitions, performances, patents, compositions, designs, research reports, software, Website content, digital or visual media, research data sets and databases all explicitly listed as valid types of academic output (REF, 2013).

Journal articles are sometimes assessed with the aid of citation analysis (e.g., in some subject areas of the UK 's Research Evaluation Framework for University research) and citation counts and publisher prestige (Torres-Salinas, Robinson-Garcia, Jimenez-Contreras, and Lopez-Corza, 2012) can be used to help assess monographs (Kousha, Thelwall, and Rezaie, 2011). Although the altmetrics movement (Priem, Groth, and Taraborelli, 2012; Priem, Piwowar, and Hemminger, 2012) is currently developing new types of impact indicators from the Web, its main focus is on journal articles and, to a lesser extent, books (e.g., Thelwall and Kousha, 2015ab, Kousha and Thelwall, 2015). The remaining output types are normally assessed by peer judgements in national research assessment exercises, and for tenure, promotion and funding applications.

This can be problematic for those producing outputs lacking quantitative indicators of value or impact and can make the evaluation of such outputs more time-consuming. As a result, it is important to assess whether any data could be gathered to inform decisions about the impact or quality of a wider variety of research outputs.

One type of non-standard academic output is the online video. Some research projects with an educational focus have produced YouTube videos and can make a case for their success by reporting download counts (Haran and Poliakoff, 2011). This statistic is problematic for videos, however, since the reach of a video may not be an indicator of its academic value (Kousha, Thelwall, and Abdoli, 2012).

Scientific datasets also have a significant role in research communication in some fields, such as medical genetics (Anagnostou, Capocasa, Milia, and Bisol, 2013; Borgman, 2012). Moreover, there has been a call for a citation-like *data usage index* to help assess the impact of scientific datasets (Ingwersen and Chavan, 2011) and Thomson Reuters (2015) has recently created the data citation index for the Web of Science that includes information about nearly two million data studies and datasets. A recent study found that only 15% of the datasets had been cited, however (Peters, Kraker, Lex, Gumpenberger, and Gorraiz, 2015), although it is not clear whether this is because few of the indexed datasets are useful in research (Wallis, Rolando, and Borgman, 2013) or because they are not being formally cited, when used. Counts of citations to the data did not have a significant positive correlation with a range of altmetrics. More fundamentally overall, impact is difficult to assess in some subject areas (Belfiore and Upchurch, 2013; Belfiore and Bennett, 2010; Brown, 2002), especially if impact is defined as generally as fundamental changes in “organisations, communities or systems” (AHRC, 2015).

Software sharing is efficient from the perspective of saving the time needed to re-develop existing applications, even when they are comprehensively described in published research. Software reuse may also help with fault-free software creation (Frakes and Kang, 2005; Mohagheghi and Conradi, 2007). The reuse of programming code, the number of downloads and user ratings may reflect the success of software projects inside or outside of academia (Rossi, Russo, and Succi, 2010; Crowston, Annabi, Howison, and Masango, 2004; Crowston, Annabi, and Howison, 2003).

Free and open source software development goes further by combining code sharing and code creating by hosting code in an environment where others can modify or add to it (Feller and Fitzgerald, 2002; Lakhani and Von Hippel, 2003; West, 2003), and some research communities use code sharing sites (Coleman and Johnson, 2014). This approach is also used in industry to some extent, although there are complex organisational consequences of its adoption (Hauge, Ayala, and Conradi, 2010). Three high profile successful free and open source software development examples are the Apache Web server, the Mozilla Web browser (Mockus, Fielding, and Herbsleb, 2002) and Linux (Hertel, Niedner, and Herrmann, 2003).

One popular code sharing site is Google Code, which began in 2006.

Developers can create a project, upload code and allow others to maintain or expand the code. Version control is particularly important when multiple coders are working together (Dabbish, Stuart, Tsay, and Herbsleb, 2012; Rodriguez-Bustos and Aponte, 2012), but Google Code provides only basic support for this. Later sites, such as GitHub, provide more sophisticated inbuilt version management. Because Google Code hosts open source projects, it also makes it easy for those wishing to reuse software from different origins within their code, promoting efficiency (Hummel, Janjic, and Atkinson, 2008).

Given the importance of software to computer science and some other research areas, indicators are needed to support evaluations of their academic or wider contributions. As a valid academic output (REF, 2013; Abramo and D'Angelo, 2015) impact indicators for software may help its creators to make a case for its value in their CVs, funding applications and for research assessment exercises. Although academic software can be associated with articles that describe it (e.g., Thelwall, Buckley, Paltoglou, Cai, and Kappas, 2010) and could be cited, software is often cited directly.

In many cases, software is also used in research and mentioned in publications without being cited, but the development of automatic methods to identify these cases may help to index such uses in the future (Pan, Yan, Wang, and Hua, 2015). The most logical indicator would be the total number of users, but this information is rarely available. If software is hosted in an online public repository, however, the total number of downloads may be used as a proxy for the number of users. Individuals may download a program multiple times or download it once and then share it with many others, but these practices seem unlikely to substantially bias download counts when compared between programs.

Research questions

The objective of this paper is to assess whether download counts can be used as indicators for the academic contributions of software. With any new proposed academic–related indicator, a logical first step is to assess whether it correlates with another indicator of better known value (Sud and Thelwall, 2014). In this case citation counts are the only available indicator for such a comparison.

The most suitable repository to analyse downloads for is Google Code. Although it closed in January 2016 (Google, 2015), it is the second most popular software repository in terms of academic citations from Scopus, with 9,705 citations in Scopus (i.e., results for a Scopus advanced search for research excellence framework ("code.google.com")). Sourceforge.net has 38,769 citations but does not report download information. The newer GitHub has only 4,329 citations and does not report download information and the other major repositories are much smaller: codeplex.com has 902 citations, Bitbucket has 314 citations, and Launchpad.net has 175 citations.

Nothing is currently known about the types of software that are most frequently used by academics. Whilst some is likely to be highly specialist, with few users, other programs may have more generic functionality and may be more widely known as a result. Information about the typical types of software used can give context to download counts so that individual programs can be interpreted relative to similar types.

Open source software often has a licence attached to it and so it is possible to assess the types of licencing used. It is helpful to know which common types of licence are applied in order to better understand how the software can be used. The following research questions drive the study.

- Considering only Google Code software cited by academic articles, do download counts have a positive correlation with citation counts?
- For Google Code software cited by academic articles, is it reasonable to use download counts as an indicator of academic (rather than commercial or other) value?
- Which types of Google Code software are most cited by academics?
- How are Google Code projects cited by academics licenced?

Methods

The first stage was to identify all Google Code program that had been cited in Scopus. The common part of the Google Code project URLs (“*code.google.com/p/*”) was entered in the “References” field (REF) of the advanced Scopus search option to identify articles citing Google Code projects. The inclusion of the /p/ part of the path excludes Google's documentation and corporate projects as well as citations to the repository itself. The reference lists of the 7,005 matching articles (68% were conference papers and 30% were articles) were then downloaded for all articles citing Google Code URLs (e.g., https://code.google.com/p/smali/). A total of 7,659 Google Code URL citations were extracted from this Scopus data using an application added to the free Webometric Analyst software (http://lexiurl.wlv.ac.uk/, see Citation menu), and manually cleaned. The number of Scopus citations to each Google Code project was then calculated by consolidating the results, producing a list of 5,370 unique repositories. For instance, the URL 'code.google.com/p/gpuocelot' had been cited 21 times.

The Google Code homepages and downloads pages of the identified projects were crawled using SocSciBot (http://socscibot.wlv.ac.uk/) and then a routine was created and added to the free software Webometric Analyst to extract information from these pages. This information included the date of the first and last code upload, the total number of downloads, and the size of the largest download. Missing pages were ignored, as were the 22 pages reporting zero downloads, although the latter did not affect the results. Only a minority of pages were still in existence, leaving 1732 code projects for analysis. Spearman correlations were used to compare citations and downloads since both are highly skewed.

To identify the types of software projects cited, the labels (if any) given by the Google Code project owners were extracted from the home pages and compared against each other to identify the most common topics. Although these labels may be created for different purposes by the various code owners, this seems like a reasonable method to get a broad overview of the software types.

Results

There is a weak but statistically significant correlation between Scopus citations and total downloads for Google Code programs (Table 1), but this may be partly due to more recently deposited software having more time both to be cited and downloaded. Although older software has longer to be cited, the correlation between the first upload date and Scopus citation counts is not statistically significant, suggesting that later software is inherently more likely to be cited than earlier software, offsetting the longer time period in which older software can be cited. To factor out time differences, the key correlations were recalculated for all software first uploaded in each year from 2008 to 2012 (Table 2). The results confirm a low but statistically significant underlying Spearman correlation of about 0.3 between downloads and citations. The size of the download is almost irrelevant for citations. It is also clear that the longer after the initial upload that software has been last updated, the more citations it attracts. Presumably, programmers are more motivated to maintain software if it is often used or cited. Conversely, better maintained software may also attract more users.

Spearman's rho	Scopus citations	Downloads	First upload date	Last upload date	Days active	Download max. size
Scopus citations	1	0.270**	0.015	0.146**	0.146**	0.079**
Downloads		1	-0.272**	0.088**	0.425**	0.112**
First upload date			1	0.609**	-0.290**	0.104**
Last upload date				1	0.476**	0.272**
Days active					1	0.273**
Download max. size						1
**Significant with p<0.01						

Table 1. Spearman correlations between indicators for 1702 programs cited at least once in Scopus and present in Google Code in March 2015. Days active records the number of days between the initial upload and the most recent upload.

Year first uploaded	Scopus citations vs. downloads	Scopus citations vs. days active	Downloads vs. days active
2008	0.195**	0.131*	0.330**
2009	0.319**	0.269**	0.406**
2010	0.299**	0.178**	0.430**
2011	0.273**	0.220**	0.369**
2012	0.358**	0.164*	0.265**
*Significant with p<0.05; **Significant with p<0.01			

Table 2. Spearman correlations between indicators for 1702 programs cited at least once in Scopus and present in Google Code in March 2015, broken down by year. Days active records the number of days between the initial upload and the most recent upload.

A scatterplot of citations against downloads suggests a weak relationship between downloads and citations although even the most downloaded software had received only one citation (figure 1). Nevertheless, the more highly cited programs tended to have attracted at least a moderate amount of citations and a positive relationship between downloads and citations is evident for software with at least 10 citations (i.e., more highly cited software is more downloaded but not vice versa).

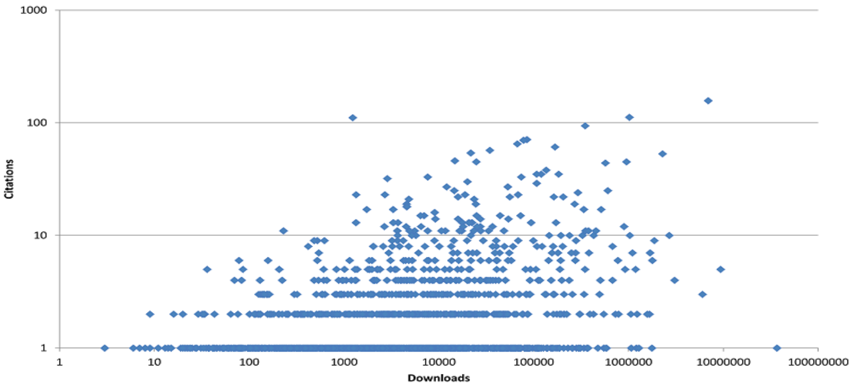
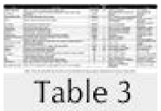


Figure 1: Scopus citations against total downloads for 1723 programs, cited at least once in Scopus and present in Google Code in March 2015 (logarithmic axis scales; articles with zero downloads are not shown).

To identify why highly downloaded software was sometimes rarely cited and why highly cited articles were relatively infrequently downloaded, individual outliers were examined. For the former, the ten most downloaded articles with only one citation were examined and for the latter, the ten articles at the top left hand of the above graph were examined.

The results (Table 3) suggest that rarely cited but heavily downloaded software tend to be general purpose utilities created by companies or independent software developers that could also be used by non-academic software developers. In contrast, the relatively highly cited but rarely downloaded programs tended to be specialist scientific code, developed by researchers and primarily or exclusively relevant to other researchers.



The program with the most substantial academic contribution may be flyspeck because it is part of an important formal mathematical proof (of the Kepler Conjecture). For this program, the moderate number of downloads clearly does not reflect its academic value.

Finally, to give some context to the results, Figure 2 shows the distribution of citation counts to, and downloads of, the articles analysed above. The distribution of citations is highly skewed: whilst most (57%) programs attract just one citation, a few programs attract hundreds. The straight line for the citations indicates a relatively pure power law but the bent downloads shape suggests a different distribution, or a mix of distributions with a power law tail (Clauset, Shalizi, and Newman, 2009).

A hooked power law or lognormal distribution is more common for counts of citations to (Thelwall and Wilson, 2014) or readers of (Thelwall and Wilson, in press) homogeneous sets of articles, and so the pure power law is unexpected and suggests a particularly strong tendency for researchers to imitate others' use of software. The different shapes between the two lines would be consistent with multiple dynamics driving the download counts, such as an academic dynamic and one from a wider user base for more general software.

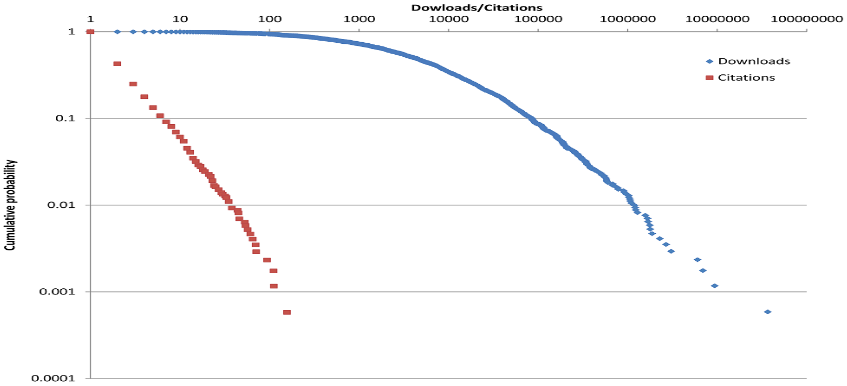


Figure 2: Downloads and Scopus citation counts for programs cited at least once in Scopus and present in Google Code in March 2015 (logarithmic axis scales).

Code labels

A total of 2208 Scopus-cited Google Code projects contained labels, including some closed projects. The number of projects and the total number of citations was counted for each label. Many of these labels described the programming languages used, with Java occurring in 20% of projects with labels and accounting for 20% of citations to projects with labels, followed by Python (13%,12%), C (5%,5%), JavaScript (5%,6%), C++ (4%,9%), CUDA (2%,4%), PHP (2%,1%), and C# (1%,1%). Other labels described operating systems, including Android (8%,12%), Linux (3%,2%), and Windows (2%,1%). There were also some generic research-related terms (academic [9%,6%], research [2%,2%], analysis [2%,2%]) and some generic computing-related terms (library [4%,5%], framework [2%,2%], algorithm [2%,1%], testing

[2%,2%]).

The most common of the remaining terms (Table 4) is Google, which seemed to be used for software created by Google rather than, for example, to reference the Google Android operating system (which seemed to be only referenced as Android).

Google Code hosts many widely downloaded and cited toolkits for technologies that Google has developed for the Web (e.g., protobuf for a data interchange format; tesseract-ocr for Optical Character Recognition (OCR), unladen-swallow for faster Python programs). Presumably Google engineers that share software would choose Google Code in preference to non-Google software sharing sites, so the presence of Google-related software is unsurprising. Moreover, although is not an academic institution, Google conducts a large amount of academic research in computing (e.g., a Scopus search for publications authored or co-authored by Google [Affiliation 'Google'] returned 3999 matches in October 2015) and so the citing of Google code in academic research is also unsurprising.

Similarly, although the core search results ranking algorithms of Google are secret, its policy of allowing engineers to spend 20% of their time on side projects (Gersch, 2013) seems likely to have generated many shared programs and the Google Code results are evidence that some of these have value in academic research.

Many of the remaining terms are related to the Web or computationally-intensive processes, such as simulation and graphics processing, and for functionality that can form a component within a larger system. From the non-Web technologies in particular it is clear that there are some niche areas within computing, and perhaps also bioinformatics and physics, for which Google Code is a useful repository.

The most cited software seems to be code that is used outside of the specialist area that created it, however. The most cited, tesseract-ocr (157 Scopus citations) is described as, 'An OCR Engine that was developed at HP Labs between 1985 and 1995... and now at Google' (https://code.google.com/p/tesseract-ocr/), and is cited in research that either applies OCR in new contexts, such as reading road signs, or as a tool within a larger system, such as a helper robot for elderly and disabled people, or Google Books (Vincent, 2007). The fourth most cited software, zxing, is a barcode scanning code library, which is also used as a component within larger systems.

Internet security is another common theme. Secure technologies use complex mathematical algorithms that are time consuming to create and shared security libraries can be an efficient way to allow new software to incorporate secure communications. The most cited security related Google Code project was Google’s browser security handbook (45 Scopus citations), which “is meant to provide Web application developers, browser engineers, and information security researchers with a one-stop reference to key security properties of contemporary Web browsers” (https://code.google.com/p/browsersec/). This is primarily a reference work although it also incorporates test software. Such information is clearly of value to people assessing the security of Web technologies.

Label (first term) and comment	Programs	Citations
Google (mainly for Google's Web technologies)	105 (4.8%)	1030 (15.4%)
Security	67 (3.0%)	364 (5.4%)
Bioinformatics	57 (2.6%)	167 (2.5%)
API (Applications Programming Interface)	51 (2.3%)	197 (2.9%)
XML (eXtensible Markup Language)	48 (2.2%)	153 (2.3%)
MATLAB (matrix laboratory) for numerical computing	44 (2.0%)	235 (3.5%)
Web	42 (1.9%)	116 (1.7%)

Simulation	41 (1.9%)	154 (2.3%)
MachineLearning	39 (1.8%)	144 (2.2%)
Simulator	38 (1.7%)	109 (1.6%)
Ajax (Asynchronous JavaScript and XML)	38 (1.7%)	77 (1.2%)
Performance	35 (1.6%)	132 (2.0%)
GPU (Graphics Processing Unit)	34 (1.5%)	281 (4.2%)
SemanticWeb	33 (1.5%)	109 (1.6%)
GWT (Google Web Toolkit)	33 (1.5%)	45 (0.7%)
RDF (Resource Description Framework) for Web metadata	32 (1.4%)	108 (1.6%)
Networking	32 (1.4%)	65 (1.0%)
Database	32 (1.4%)	55 (0.8%)
Ontology	31 (1.4%)	163 (2.4%)
OWL (Web Ontology Language)	31 (1.4%)	117 (1.7%)
Arduino – commercial open-source electronics platform	30 (1.4%)	52 (0.8%)
OpenGL (Open Graphics Library)	30 (1.4%)	51 (0.8%)
Physics	29 (1.3%)	89 (1.3%)
Visualization	28 (1.3%)	138 (2.1%)
Modeling	28 (1.3%)	78 (1.2%)

Table 4. The 25 most common labels in cited Google Code projects, excluding programming languages, operating systems and generic computing and research terms.

Licence types

No licence type is dominant, with the most popular being the Apache license 2.0 (Table 5). Although the projects in Google Code are open source, this shows that their use has some restrictions and that the software developers have different needs for their software. This may reflect ideological differences between the code owners or differing types of sharing or commercial exploitation needs.

The Apache licence 2.0 was created in 2004 by the Apache software foundation, a charitable foundation in the USA for creating shared programs, as a convenient way for programmers to licence their code by citing the licence URL rather than by composing their own text or copying the licence test. It grants “a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form” (http://www.apache.org/licenses/LICENSE-2.0), which essentially allows anyone to use the software in any way, although they must acknowledge its origins and include a disclaimer. The new Berkeley software distribution license has similar affordances.

The second most popular type of licence is arguably more ideological because it blocks uses of the software that are not shared. GNU general public license v3, for example, continues a long tradition of supporting computer scientists that want to share their work but retain some control over how it is reused (Rychlicki, 2008; Sauer, 2007). Copyleft requires any derivative programs to be distributed free and hence blocks software that is sold from incorporating the code. This is in general public license v3 but not general public license v2, although general public license v3 allows commercial developments as long as the software itself is not sold (Rychlicki, 2008).

Licence	Projects	URL
Apache License 2.0	369 (21.4%)	http://www.apache.org/licenses/LICENSE-2.0
Artistic License/GPL	16 (0.9%)	https://gnu.org/licenses/gpl.html
Eclipse Public License 1.0	22 (1.3%)	https://www.eclipse.org/legal/epl-v10.html

GNU GPL v2	250 (14.5%)	http://www.gnu.org/licenses/gpl-2.0.html
GNU GPL v3	348 (20.2%)	http://www.gnu.org/copyleft/gpl.html
GNU Lesser GPL	197 (11.4%)	https://www.gnu.org/licenses/lgpl.html
MIT License	174 (10.1%)	http://opensource.org/licenses/MIT
Mozilla Public License 1.1	18 (1%)	https://www.mozilla.org/MPL/1.1/
Multiple Licenses	1 (0.1%)	
New BSD License	284 (16.5%)	http://opensource.org/licenses/BSD-3-Clause
Other Open Source	42 (2.4%)	
Public domain	2 (0.1%)	
Total	**1723 (100%)**	

Table 5. The licences declared for the Google Code projects.

Discussion and limitations

An important limitation of this study is that it only covers one software code repository. It seems unlikely that the results would be fundamentally different for others, such as SourceForge, GitHub and BitBucket, if they included download statistics. Nevertheless, there are likely to be differences due to the average age of the software in each one, the different facilities available and the presence or absence of major contributors, such as Google. A study of software located on university Websites might generate much stronger correlations between citations and downloads, if available, due to fewer general purpose programs. For example, the two programs used in the current paper are hosted on academic Websites and can be found by cited reference searches [REF("lexiurl.wlv.ac.uk") gets seven citations and REF("socscibot.wlv.ac.uk") gets six citations].

Another limitation is that software may have been recently transferred from Google Code to another repository, as have all of Google's own projects (Google, 2015), and so its Google Code downloads would underestimate the total usage. Google Code's Scopus citations peaked in 2013 (1,746, compared to 1,586 in 2014), confirming a shift away from it. Perhaps most significantly, however, the study ignored all software in Google Code that had not been cited at least once in Scopus. Including this software with zero citation counts in the data set could alter the correlations, although it is not clear whether they would have increased or decreased.

Presumably the vast majority of Google Code software does not target an academic audience and so selecting just the programs with academic citations is a convenient way of identifying an academic-related subset. Nevertheless, there is an unknown number of uncited programs that target an academic audience in Google Code. Given the power law relationship found for the cited software, it would be reasonable to expect this set to be very large, and perhaps larger than all the cited software. It seems probable that software targeted by academics but uncited would be less likely to be downloaded frequently than cited software and so the overall correlation might increase, but there is no evidence to test this hypothesis.

A limitation of the investigation of labels used to describe the cited software is that these vary in terms of generality and so the most popular labels tend to be the most general terms used as well as terms describing the software language or operating system used. This may have obscured some themes in the software, such as a range of similar types of specialist use that were described with different terms.

An important practical limitation is that download counts are not available for some open source software repositories, such as SourceForge.net and GitHub, which may soon be the most popular. GitHub reports publically the number of users that register to watch a project or that award it with a star and these could be used for alternative indicators.

Another practical limitation is that many people contribute software to collaborative projects, such as Linux, and their contribution cannot be directly cited. In addition, useful general purpose software code may not be cited even when it makes a substantial contribution to a study. Software may also be cited indirectly. For example, the collaboratively authored open source statistical package R contains many packages created by individual named authors and these may be cited via the package documentation or any describing article (Calenge, 2006) rather than the code URL.

Conclusions

Although Scopus citations to software correlate significantly and positively with total Google Code downloads, the correlations are low, at about 0.3. Thus, there is a weak tendency for more used software to be more cited. The low correlation seems to be due to the data set mixing academic software for a niche academic audience and general purpose software that is more widely useful to software developers. Thus, whilst download counts can be presented by academic software developers as evidence of the value of their work, download counts should not be directly compared between computer programs if one targets a more specialist audience (e.g., academics in a specific field) than another (e.g., all Website developers). Instead, the download count should be presented to support a self-contained claim for the utility of the software. Of course, download counts are easy to manipulate by the author repeatedly downloading their own code and there does not seem to be a way to detect this and so evaluators should use their judgement to decide whether a reported download count is reasonable or not. Authors may also wish to use the download counts to self-evaluate the uptake of their software (see also: Wouters and Costas, 2012). This could be particularly valuable for those producing successful software since they may be encouraged to upgrade it or to pursue related work.

The strong power law in the distribution of citations to software suggests a significant amount of imitation between academics, with researchers being more prepared to use a program if others are already using it. This may be due to the publicity or endorsement given by new users to software by citing it or there may be a feedback loop in which programmers continue to maintain and improve software when it is used.

The presence of Google-engineered code in the results is also evidence that the software company has generated a substantial amount of freely-shared software that is valuable for academic research. The analysis of the labels also points to the usefulness of shared code as a component in larger systems and especially as an efficient way to incorporate complex code, such as that dealing with graphics, image processing, or security.

The shared software is primarily licenced either to allow essentially any uses to be made of it, or to restrict uses that are not freely shared (copyleft). This seems to be an ideological distinction that impacts primarily on commercial uses, although both types of licence are common. Overall, however, the shared software is licenced in a way that is altruistic in the sense of not bringing direct commercial benefits to the originator.

In terms of future work, finding out more about the types of software that are used by researchers would enable guidelines to be built to encourage the sharing of useful code. It would also be interesting to systematically identify the range of ways in which software is shared and in which shared software is acknowledged in others' work so that a more comprehensive study could assess the overall contribution of software sharing to academic research. Finally, it is important to assess the perspective of the users in order to interpret both download counts and citations to software in a broader context.

About the authors

Mike Thelwall is the head of the Statistical Cybermetrics Research Group, University of Wolverhampton, Wulfruna Street, Wolverhampton WV1 1LY, U.K. He has developed a wide range of software for gathering and analysing Web data, including hyperlink analysis, sentiment analysis and content analysis for Twitter, YouTube, blogs and the Web in general. He can be reached at m dot thelwall at wlv.ac.uk
Kayvan Kousha is a researcher in the Statistical Cybermetrics Research Group at the University of Wolverhampton, UK. His research includes Web citation analysis and online scholarly impact assessment using Webometrics and altmetrics methods. He can be contacted at k.kousha1 [....at....] wlv. ac. uk

Abramo, G. & D'Angelo, C. A. (2015). The VQR, Italy's second national research assessment: methodological failures and ranking distortions. *Journal of the Association for Information Science and Technology*, *66*(11), 2202-2214.

Arts and Humanities Research Council. (2015). Understanding your project: a guide to self-evaluation. London: Arts and Humanities Research Council. Retrieved from http://www.ahrc.ac.uk/What-We-Do/Build-the-evidence-base/Pages/Self-evaluation.aspx. (Archived by WebCite® at http://www.Webcitation.org/6X9D5rdhC)

Anagnostou, P., Capocasa, M., Milia, N. & Bisol, G. D. (2013). Research data sharing: lessons from forensic genetics. *Forensic Science International: Genetics*, *7*(6), e117-e119.

Belfiore, E. & Upchurch, A. (Eds.). (2013). *Humanities in the twenty-first century: beyond utility and markets*. Basingstoke, UK: Palgrave Macmillan.

Belfiore, E. & Bennett, O. (2010). Beyond the “Toolkit Approach”: arts impact evaluation research and the realities of cultural policy making. *Journal for Cultural Research*, *14*(2), 121-142.

Borgman, C. L. (2012). The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology*, *63*(6), 1059-1078.

Brown, C. D. (2002). Straddling the humanities and social sciences: the research process of music scholars. *Library & Information Science Research*, *24*(1), 73-94.

Calenge, C. (2006). The package “adehabitat” for the R software: a tool for the analysis of space and habitat use by animals. *Ecological modelling*, *197*(3), 516-519.

Clauset, A., Shalizi, C. R. & Newman, M. E. (2009). Power-law distributions in empirical data. *SIAM review*, *51*(4), 661-703.

Coleman, R. & Johnson, M. A. (2014). A study of Scala repositories on GitHub. *International Journal of Advanced Computer Science and Applications*, *5*(7), 141-148.

Crowston, K., Annabi, H., Howison, J. (2003). Defining open source software project success. In *Proceedings of the 24th International Conference on Information Systems, Seattle, Washington, USA*. (pp. 327-340). Atlanta: GA: Association for Information Systems. Retrieved from https://surface.syr.edu/cgi/viewcontent.cgi?article=1080&context=istpub (Archived by WebCite® at http://www.webcitation.org/6fUr99qtg)

Crowston, K., Annabi, H., Howison, J. & Masango, C. (2004). *Towards a portfolio of FLOSS project success measures.* Paper presented at the Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering, Edinburgh, Scotland, UK, 25 May. Retrieved from http://surface.syr.edu/cgi/viewcontent.cgi?article=1077&context=istpub. (Archived by WebCite® at http://www.webcitation.org/6fUqeoLOy)

Frakes, W.B. & Kyo Kang, (2005). Software reuse research: status and future. *IEEE Transactions on Software Engineering*, *31*(7), 529-536.

Dabbish, L., Stuart, C., Tsay, J. & Herbsleb, J. (2012). Social coding in GitHub: transparency and collaboration in an open software repository. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, Seattle, Washington, USA* (pp. 1277-1286). New York, NY: ACM.

Feller, J. & Fitzgerald, B. (2002). *Understanding open source software development*. London: Addison-Wesley.

Gersch, K. (2013). Google's Best New Innovation: Rules Around '20% Time'. *Forbes Magazine*. Retreived from http://www.forbes.com/sites/johnkotter/2013/08/21/googles-best-new-innovation-rules-around-20-time/ (Archived by WebCite® at http://www.Webcitation.org/6fImEhjT9)

Google (March 12, 2015). Bidding farewell to Google Code. [Web log entry]. Retrieved from http://google-opensource.blogspot.co.uk/2015/03/farewell-to-google-code.html. (Archived by WebCite® at http://www.Webcitation.org/6X9DEkmSI)

Haran, B. & Poliakoff, M. (2011). The periodic table of videos. *Science*, *332*(6033), 1046-1047.

Hauge, Ø., Ayala, C. & Conradi, R. (2010). Adoption of open source software in software-intensive organizations–a systematic literature review. *Information and Software Technology*, *52*(11), 1133-1154.

Hertel, G., Niedner, S. & Herrmann, S. (2003). Motivation of software developers in open source projects: an Internet-based survey of contributors to the Linux kernel. *Research policy*, *32*(7), 1159-1177.

Hummel, O., Janjic, W. & Atkinson, C. (2008). Code conjurer: pulling reusable software out of thin air. *IEEE Software*, *25*(5), 45-52.

Ingwersen, P. & Chavan, V. (2011). Indicators for the data usage index (DUI): An incentive for publishing primary biodiversity data through global information infrastructure. *BMC Bioinformatics*, *12*, (Suppl 15): S3.

Kousha, K., Thelwall, M. & Abdoli, M. (2012). The role of online videos in research communication: a content analysis of YouTube videos cited in academic publications. *Journal of the American Society for Information Science and Technology*, *63*(9), 1710–1727.

Kousha, K., Thelwall, M. & Rezaie, S. (2011). Assessing the citation impact of books: the role of Google Books, Google Scholar and Scopus. *Journal of the American Society for Information Science and Technology*, *62*(11), 2147–2164.

Kousha, K. & Thelwall, M. (2015c). Web indicators for research evaluation, part 3: books and non-standard outputs. *El Profesional de la Información*, *24*(6). http://www.elprofesionaldelainformacion.com/contenidos/2015/nov/04.pdf (Archived by WebCite® at http://www.Webcitation.org/6fImSy9fv)

Lakhani, K. R. & Von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, *32*(6), 923-943.

Mockus, A., Fielding, R. T. & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, *11*(3), 309-346.

Mohagheghi, P. & Conradi, R. (2007). Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*, *12*(5), 471-516.

Pan, X., Yan, E., Wang, Q. & Hua, W. (2015). Assessing the impact of software on science: a bootstrapped learning of software entities in full-text papers. *Journal of Informetrics*, *9*(4), 860-871.

Peters, I., Kraker, P., Lex, E., Gumpenberger, C. & Gorraiz, J. (2015). *Research data explored: citations versus altmetrics*. Retrieved from http://arxiv.org/pdf/1501.03342v2.pdf

Priem, J., Groth, P. & Taraborelli, D. (2012). The altmetrics collection. *PLOS ONE*, *7*(11), e48753. Retrieved from http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0048753 (Archived by WebCitereg; at http://www.webcitation.org/6fUlEpPlw)

Priem, J., Piwowar, H. A. & Hemminger, B. M. (2012). *Altmetrics in the wild: using social media to explore scholarly impact.* Retrieved from http://arxiv.org/html/1203.4745v1?

Research Excellence Framework. (2013). Output information requirements. Retrieved from http://www.ref.ac.uk/about/guidance/submittingresearchoutputs/. (Archived by WebCite® at http://www.Webcitation.org/6X9DIhewb)

Rodriguez-Bustos, C. & Aponte, J. (2012). How distributed version control systems impact open source software projects. In *Proceedings of the 9th IEEE Working Conference on Mining Software*

Repositories (pp. 36-39). Los Alamitos, CA: IEEE Press.

Rossi, B., Russo, B. & Succi, G. (2010). Download patterns and releases in open source software projects: a perfect symbiosis? In Pär Ågerfalk, Cornelia Boldyreff, Jesús M. González-Barahona, Gregory R. Madey & John Noll, (Eds.). *Open source software: new horizons. Proceedings 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010, Notre Dame, IN, USA, May 30 – June 2, 2010* (pp. 252-267). Berlin: Springer. (IFIP Advances in Information and Communication Technology Volume 319, p. 252-267) Retrieved from http://link.springer.com/chapter/10.1007%2F978-3-642-13244-5_20#page-1.

Rychlicki, T. (2008). GPLv3: New software licence and new axiology of intellectual property law. *European Intellectual Property Review, 30*(6), 232.

Sauer, R. M. (2007). Why develop open-source software? the role of non-pecuniary benefits, monetary rewards, and open-source licence type. *Oxford Review of Economic Policy*, *23*(4), 605-619.

Sud, P. & Thelwall, M. (2014). Evaluating altmetrics. *Scientometrics, 98*(2), 1131-1143.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D. & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, *61*(12), 2544-2558.

Thelwall, M. & Kousha, K. (2015a). Web indicators for research evaluation, part 1: citations and links to academic articles from the Web. *El Profesional de la Información*, *24*(5), 587-606.

Thelwall, M. & Kousha, K. (2015b). Web indicators for research evaluation, part 2: social media metrics. *El Profesional de la Información, 24*(5), 607-620.

Thelwall, M. & Wilson, P. (2014). Distributions for cited articles from individual subjects and years. *Journal of Informetrics, 8*(4), 824-839.

Thelwall, M. & Wilson, P. (in press). Mendeley readership altmetrics for medical articles: an analysis of 45 fields, *Journal of the Association for Information Science and Technology*.

Thomson Reuters (2015). The data citation index. Retrieved from http://wokinfo.com/products_tools/multidisciplinary/dci/. (Archived by WebCite® at http://www.Webcitation.org/6X9DOefxN)

Torres-Salinas, D., Robinson-Garcia, N., Jimenez-Contreras, E. & Lopez-Corza, E.D. (2012). Towards a book publishers citation reports: first approach using the 'book citation index'. *Revista Espanola De Documentacion Cientifica, 35*(4), 615-624.

Vincent, L. (2007). Google book search: document understanding on a massive scale. In *Ninth International Conference on Document Analysis and Recognition* (pp. 819-823). Los Alamitos, CA: IEEE Press

Wallis, J. C., Rolando, E. & Borgman, C. L. (2013). If we share data, will anyone use them? Data sharing and reuse in the long tail of science and technology. *PloS ONE*, *8*(7), e67332. http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0067332 (Archived by WebCite® at http://www.webcitation.org/6fUqOFmVj)

West, J. (2003). How open is open enough?: melding proprietary and open source platform strategies. *Research Policy*, *32*(7), 1259-1285.

Wouters, P. & Costas, R. (2012). Users, narcissism and control: tracking the impact of scholarly publications in the 21st century. Utrecht: SURF foundation. Retrieved from http://www.surffoundation.nl/nl/publicaties/Documents/Users%20narcissism%20and%20control.pdf (Archived by WebCite® at http://www.Webcitation.org/6X9DS0IQd)

Find other papers on this subject

Scholar Search | Google Search | Bing

Check for citations, using Google Scholar

0 Tweet

1

Contents | **Author index** | **Subject index** | **Search** | **Home**