Draft version January 9, 2018
Typeset using LaTeX **modern** style in AASTeX61

SCHROEDINGER'S CODE: A PRELIMINARY STUDY ON RESEARCH SOURCE CODE AVAILABILITY AND LINK PERSISTENCE IN ASTROPHYSICS

Alice Allen,[1, 2] Peter J. Teuben,[2] and P. Wesley Ryan[1]

[1] *Astrophysics Source Code Library, ascl.net*

[2] *University of Maryland, Department of Astronomy, PSC 1208, 4254 Stadium Drive, College Park, MD 20742*

ABSTRACT

We examined software usage in a sample set of astrophysics research articles published in 2015 and searched for source code for the software mentioned in these research papers. We categorized the software to indicate whether source code is available for download and whether there are restrictions to accessing it, and if source code is not available, whether some other form of the software, such as a binary, is. We also extracted hyperlinks from one journal's 2015 research articles, as links in articles can serve as an acknowledgment of software use and lead to data used in the research, and tested them to determine which of these URLs are still accessible. For our sample of 715 software instances in the 166 articles we examined, we were able to categorize 418 records as to availability of source code and found that 285 unique codes were used, 58% of which offer source code available online for download. Of the 2,558 hyperlinks extracted from 1,669 research articles, at best, 90% of them were available over our testing period.

Corresponding author: Alice Allen
aallen@ascl.net

1. INTRODUCTION

"Is a paper for which the author feels that it is not worth making the code available worth reviewing?" (Vitek & Kalibera 2011)

Astrophysics is, like nearly every discipline (Ludaescher et al. 2016; Howison et al. 2015; Hettrick et al. 2014), dependent on software for many aspects of the research process (Momcheva & Tollerud 2015), from gathering, reducing, and cleaning data, to modeling and visualizing data and creating analyses of enormous amounts of data (Soito & Hwang 2016; Brown et al. 2007). Reliance on software cannot be overstated, yet as computational methods have proliferated, research has become less transparent, reproducible, and falsifiable because these methods have not been shared or made available along with the research they enable (Baker 2016; Goble et al. 2016; Stodden et al. 2016; Marwick 2015; Shamir et al. 2013; Morin et al. 2012).

Certainly as these methods started to be used, it was inconvenient to share long or complex codes if they could not be included within research papers; sharing software was typically limited to those within close physical proximity of the software author. As technology has advanced, sharing source code with researchers everywhere is possible; still, practices have not caught up with what technology now enables. Reasons for this are varied and have been covered in the literature (Ince et al. 2012; Barnes 2010; Stodden 2010; Weiner et al. 2009), in sessions at astronomy meetings (Teuben et al. 2014; Allen et al. 2013) and more broadly at multidisciplinary workshops (Goble et al. 2016; Ahalt et al. 2015; Katz et al. 2014).

We will not discuss the reasons here except to acknowledge they exist, and that the literature, meeting and workshop reports that cover barriers and reluctance to sharing also include calls for greater openness and availability of source code. Indeed, some have called for peer review of research software (Joppa et al. 2013) and have worked to implement it (Smith et al. 2017; Limare 2012). Organizations and events dedicated to improving scholarly communication and free access to research artifacts are actively working to increase transparency and reproducibility (Force 11 (Smith et al. 2016), OpenCon[1], Workshop on Sustainable Software for Science: Practice and Experiences[2], Software Sustainability Institute[3]) and government agencies are requiring that research products paid for by taxpayers, particularly data and software, be available to them (NASA 2014). Journal practices have evolved and may request, or even require, that source code and data be made available, as is seen with *Nature* and *Science*, and Elsevier has declared a new "academic content class" called Original Software Publications, which requires peer review of the software in that content class. Domain and general journals focused on peer review of software also

[1] `http://www.opencon2017.org/`
[2] `http://wssspe.researchcomputing.org.uk/`
[3] `https://www.software.ac.uk/`

exist, such as *Source Code for Biology and Medicine*[4], *Image Processing On Line*[5], and *Journal of Open Source Software*[6].

Infrastructure and services such as the NASA Astrophysics Data System[7] (ADS), GitHub[8], the Astrophysics Source Code Library[9] (ASCL), Zenodo[10], and Figshare[11] make it easier to share and discover useful research software. With more ways to make these programs available for examination, we wonder: How many codes that are used and clearly identified in research can be examined by other researchers? And how much research is enabled by software without it being made explicitly clear in an article that a computational method was used? These are questions we are starting to research.

We are not the first to consider these questions. A wide-ranging project undertaken by Collberg & Proebsting (2016) included studying papers by Computer Systems researchers to find those with results dependent on software and to get the source code if possible. Among other questions, they also studied whether the software would build within 30 minutes, or at all, without help from the software authors, whether National Science Foundation (NSF) funding has an impact on sharing source code, and examined results from other studies on research software availability. In their own study, out of the articles they examined that were backed by software, "for 56.2% of them the authors were able to make that [source] code available."

In a robust study of biology articles, Howison & Bullard (2016) examined software citations and accessibility, and found that "software is frequently inaccessible" and that "only between 24%-40% provide source code." Their study covered whether code could be accessed, whether software was licensed in a way to permit reuse, how well authors provided software creators with credit for these scientific contributions, and also whether version information was provided in the articles.

Here we report on our preliminary efforts on the first question, which we undertook to find out how we might do a comprehensive study to answer this and related questions. The availability of hyperlinked resources (especially code and data) related to published research is also important for the reproducibility, reliability, and falsifiability of scientific results, so we also examined the extent to which hyperlinked resources are accessible.

Unlike Howison & Bullard (2016) and Collberg & Proebsting (2016), we restricted our study to the issue of transparency of the computational methods – can the reader see what was done? – which certainly includes the ability to identify the software and find it, but we did not tackle the other clearly important issues of how well codes were cited, how they were licensed, whether they would build, nor whether the version used in an article was still available in this preliminary study. We also sought to determine

[4] https://scfbm.biomedcentral.com/
[5] urlhttp://www.ipol.im/
[6] http://joss.theoj.org/
[7] http://adsabs.harvard.edu/
[8] https://github.com/
[9] http://ascl.net/
[10] https://zenodo.org/
[11] https://figshare.com/

the persistence of hyperlinks included in research papers, as such links are often for software and data needed to ensure the transparency of the research.

2. ISSUES IN ANSWERING THE RESEARCH QUESTIONS

Several issues arise in trying to conduct this type of study. Research articles do not always include what software was used; some do not even acknowledge that software was used, yet some tasks described are too cumbersome to not have had some sort of software involved. Software often unintentionally gets left out, as in this text:

> With the observed IR PSD, we simulate light curves using the method of Timmer & Koenig (1995). Gaps identical to the IR light curve were artificially introduced in the simulated light curves and the resulting PSD is compared to the PSD of the original light curve. In the second test, we introduce gaps in the RXTE X-ray light curve (which does not have any gaps) identical to the simultaneous IR light curve, fill these gaps using similar technique as in the IR data, and compare the PSDs. (Kalamkar et al. 2016)

"... fill these gaps using similar technique as in the IR data..." In other words, a code was used (pyLCSIM, Campana (2017)), though it is not mentioned by name. Maybe it is obvious to others that some unnamed code was referred to in that phrase, but it was not obvious to the first author when she first ran across this text.

It is easy to determine a paper used research software when the text says, "The photometric reduction was performed with AstPhot, a synthetic aperture photometry tool" (Tubiana et al. 2015), or, "In our calculations we made use of the public code developed by Dany Page, NSCool" (Bonanno & Urpin 2015).

Or even when the software used is not named, as in Santos-Sanz et al. (2015), which said "A common reduction software programmed in IDL was used for the photometry of all the previously calibrated images." We do not know what software was used, but we do know something was.

One of our longer term motivations for doing this work is to refine our ability to search for code use programmatically, and as we continue to develop the data we have gathered, we will use it to further train and benchmark our automated method. One of the difficulties in searching, however, is when the software is not mentioned by name in text, such as "Original light curves were processed and corrected from phenomenological effects such as outliers, jump and drifts according to the method of García et al. (2011)." (Vrard et al. 2015) or "We performed ... data reductions following the procedures outlined in Taddia et al. (2013b)." (Taddia et al. 2015).

Though we have selected specific examples to demonstrate some of the challenges in trying to determine the availability of software used in research, we certainly do not mean any disrespect to any author (of article or software) or journal. These examples follow norms in research publishing (clearly there is nothing wrong with referring the reader elsewhere for background or more information!) but still present

Table 1. ADS Bumblebee searches
Total papers for the given years[a]

Journal	2015	2010
ApJ	3,055	2,539
MNRAS	3,140	1,987
A&A	1,805	1,932

[a]Information retrieved 2017 June 4

a challenge to finding computational methods. We want to shed a light on one aspect of science – software use – that has an impact on the transparency and replicability of astrophysics, and we do not fault any author for following standard practices and discipline norms.

3. METHODS

3.1. *Planning*

We are considering examining software use in papers over time. Though the Astrophysics Source Code Library (ASCL) was founded in 1999, it started being indexed by the NASA Astrophysics Data System (ADS) only in January 2012. One impetus for conducting such research is to see what impact the ASCL and the other efforts to improve source code availability mentioned in the Introduction may have on the community. We think five-year intervals are broad enough to show changes in source code accessibility without being too onerous to do, and selected 2010 as a good start for the cycle of five-year intervals for several reasons. First, it's a nice round number, but more importantly, it would give us data on software availability before the indexing of the ASCL by ADS began; this is also before many other efforts and organizations with similar concerns were started or could have an impact, including Force11, Workshop on Sustainable Software for Science: Practice and Experiences, and the Software Sustainability Institute, and also before NASA and other funding agencies required data (including software) management plans.[12]

We searched ADS Bumblebee to learn how many papers each of three journals, the *Astrophysical Journal*, *Astronomy and Astrophysics*, and *Monthly Notices of the Royal Astronomical Society* (*ApJ*, *A&A*, and *MNRAS* respectively), published in 2010 and 2015 (see Table 1).

ApJ, *MNRAS*, and *A&A* collectively published 8,000 papers in 2015 and 6,458 papers in 2010. We eventually want to look at software use in 2010, but for this pre-

[12] `https://www.nasa.gov/open/researchaccess/frequently-asked-questions#dmpfaqs`

liminary study, we chose to start our research with papers published in 2015 to gather information on recent research software availability. Using articles from roughly the same time period as studies in other disciplines makes it feasible to compare our results with those other studies and other researchers may find this study useful for that reason.

Though our plans are to extend this research to include *ApJ* and *MNRAS* articles, each of these presented special concerns for our initial foray into answering our research questions. In 2015, *MNRAS* had one editor in particular who was very active in promoting software citation and open access to code; additionally, these papers are still behind a paywall. *ApJ*, though offering free access to 2015 articles, did not officially allow publication of software papers nor formal software citation until 2016, though, wisely, these rules were not rigorously enforced and software papers and citations for codes certainly appear in its 2015 issues. *A&A* fell between these two; in 2015, it published software papers and allowed software citation in an ADS-capturable way but (so far as we know) did not have an editor who was really pushing to improve the journal in this way as *MNRAS* did. We felt papers from *A&A* would offer a balanced picture. Also, the journal offered a reasonable rather than overwhelming number of articles and it was relatively easy to download the year's worth of papers to examine. The papers are free access and readily available to anyone who would like to check our research (which we encourage). We therefore conducted this preliminary study using 2015 *A&A* articles.

3.2. *Article selection*

There were 1,805 articles published in *A&A* in 2015. We removed the 131 Letters from consideration. This left 1,653 research articles and 21 Errata. We selected 10% of these remaining articles for examination by ordering the papers by article number (an identifier given by the publisher) and selecting every 10th paper, giving us a sample set of 166 papers to examine.

The distribution of selected papers by publication month varied from a low of nine for October (out of 127 articles that month) to a high of 20 for November (out of 143 articles that month). November's issue included a special focus on Rosetta; perhaps focus articles had a firm due date, resulting in an abundance of consecutively-numbered articles for that issue.

Three errata were among the selected articles; we replaced these with the original papers in two cases, as the original articles were published in 2015; the remaining erratum was for a paper from 2014 so we replaced the erratum with the paper immediately after it in the ordered list. Although anyone following the procedures above should select the same articles we used, a table of article IDs and links to the PDFs of those we chose is in Appendix C.

3.3. *Looking for code in all collected papers*

Table 2. Search terms used to find code methods

code	python	numeric	routine	script	fortran	program
http	model	software	librar	algorithm	IDL	ftp
pipeline	github	procedure	iraf	package	comput	recipe

We skimmed each article from start to end for evidence of software use. As Howison & Bullard (2016) did, we looked not just for software indicated by a formal citation, but also for software "mentions", as they put it, to include codes in text, figures, footnotes, and appendices. Unlike Howison & Bullard, we also captured text that indicated software was very likely used though was not specified.

We considered what to include or perhaps exclude. Our focus is on source code; we weren't sure initially whether we wanted to find software in all its incarnations – binaries, web calculating services, commercial code mentions, software written in the 1980s and before – and then sort through whatever we find, or gather just source code. In the end, we excluded two large Government-run software systems (NOAA's Global Forecast System and the Comprehensive Large-Array-Data Stewardship System) and computational methods that are essentially "astro/scientific arithmetic" (as we came to think of them) or manipulations, such as Fourier transform calculations, but otherwise included everything else. We looked specifically for mentioned use of software; if an author thought a script was important enough to mention, we counted it. We admit that where to draw the line on what to include is somewhat arbitrary and may depend on one's specific area of research, as what is common in one particular domain may not be used in another, and that others doing similar research may draw the line in a different place.

After our initial read, we did a full-text search of each of the 166 article PDFs for a minimum set of search terms, listed in Table 2, to try to ensure no computational methods were overlooked.

Additional search terms were used occasionally for specific papers when the read-through of an article suggested such searching might be useful.

We copied text that indicated definite or possible use of computational methods from the article to create a datafile with 166 records, one for each paper.

As a quality check for the search method, eight articles, 4.8% of our sample chosen for this preliminary work, were examined by others (members of ASCL's Advisory Committee) and their results compared against the first author's.

We went through the collected text to organize and atomize the information. The 166 papers yielded a dataset of 715 records, with each record representing possible use of one computational method.

In a first pass of these data, we identified software registered in the ASCL, though we did not catch all such entries. In our second pass, we looked to see which records identified software by name or had a URL or other very definite indication for software that was readily searchable. Subsequent passes resulted in increasingly cleaner data and deletion of entries found to be not software related. Of our 715 records, 418 of them were ultimately used for the research discussed in the rest of this article.

The records we did not include in this study do not offer readily searchable information for software; some, for example, refer the reader to another paper or papers or may state that a code method was used but offer no other information about it. Other records are for what we refer to as "hidden software," which is where article text strongly suggests that a computational method had to have been used, but does not provide any information whatsoever as to what the method was. We are interested in trying to determine how much research software is hidden so collected these data in addition to information that is more transparent about software use; we hope to use these entries in future research but they are outside the scope of this article so are not considered here.

We looked for download sites for the software in the 418 records by doing the following:

1. Check to see whether the code had an ASCL entry
2. Look in the examined paper for URL or other location information; if the code had a citation, in the paper cited for a URL or other location information
3. Use Google to search for the name of the code, its author(s), and other relevant information
4. Look on the author's and/or project website

We took an additional step for some software; we sent one of the ASCL's standard emails asking whether there was a download site for the source code to the software author(s), and in one case, one article author, with the intent of registering the software in the ASCL if the code was available.

3.4. *Categorizing availability*

Once we had completed our search for a download site for a code, we categorized the code's availability with one of the designations shown in Table 3.

Some software packages are contained within a larger package, such as NuSTARDAS in HEASoft (NASA High Energy Astrophysics Science Archive Research Center (Heasarc) 2014), and were attributed to the larger package and categorized under that larger package. In other words, if a paper stated it used NuSTARDAS, the software is listed as HEASoft for categorization. There are exceptions to this, such as DAOPHOT (Stetson 1987), which was attributed to IRAF (Tody 1986, 1993) when a paper

Table 3. Category explanations

Category	Explanation
A	We found source code available for download without reservation or impediment.
B	We found an executable file, such as a binary or other compiled file, available for download without reservation or impediment.
C	We learned from correspondence, website, or other authority that the source code is available only to collaborators.
N	We did not find a way to download the source code. This could be for several reasons, such as no download site exists, one exists but we were unsuccessful in finding it, or a site for the code exists but the URL for the download did not work when we tried it.
W	The code can be used as an online service, but we could not find a download site for the source code. In cases where the software is available for use as an online service and the source code is available for download without reservation or impediment, the code was categorized as A; if available as a binary and a web service, the code was categorized as B.
GS	The source code is available for download behind a soft gate, which means one must provide information of some kind online to get immediate automated access to download the code.
GH	The source code is stated to be available and is behind a hard gate that requires human action of some kind to receive the source code, such as an email to the author, an online form without immediate automated access, or a requirement to attend training before gaining access to the source code.
P1	The source code itself is available for purchase (such as *Numerical Recipes* codes).
P2	The software is commercial software purchased as an executable (such as Feko, Mathematica, and IDL).
O	Anything not fitting a designation listed above.

stated it had used IRAF, but is listed separately when it appeared alone. We recognize that it is somewhat unfair to some packages to do this, but as we are primarily determining code availability, not specific package, routine, or function use, have made this pragmatic and in practice occasionally arbitrary decision.

We initially categorized all records for codes in the ASCL as *A*, and then tested the availability of these packages by verifying that the links the ASCL has for these codes worked, and where they did not, found a new download link, or failing to find one, adjusted the categorization as needed.

Some entries needed close examination, such as when a link was retrieved that went to a site that offered both data and software. If only the data from the site were used in the paper under examination, we did not include it in our analysis. In other words, only when the authors of a selected paper used a code on a site did we include it. One such case is MILES; though software on that site generated available data, a researcher may use only the generated data and not run any code from that site herself.

Algorithms obviously can be implemented in different languages and with different coding techniques. The Lomb-Scargle periodigram algorithm (Lomb 1976; Scargle 1982) is widely used and multiple implementations of it exist[13]; not all articles using it specified which implementation was used. We scored this as *P1* when a *Numerical Recipes* (Press et al. 1986, 1992a,b, 2002) implementation was specified, *W* when an online service was used, and the unknown cases as *N*. We struggled with this a bit, but as said above, we are determining code availability, and if we do not know the implementation used, we cannot examine that specific software.

We emailed some authors when we did not find a download site for a code through other methods using the ASCL's software request email template (Appendix B); this is a standard practice for the ASCL editors when we are unable to find a download site for software we come across in papers, as we hope to register the code in the ASCL. We sent 46 emails seeking information on code availability in our data file for 49 unique codes and received 19 replies.

We started registering software that meets ASCL criteria as we found it in working through the records, but had to abandon doing so for the duration of our research period due to time and will add these codes at a later date.

3.5. *Checking links in papers*

As mentioned in the Introduction, hyperlinks in articles may point to software or data, informing the user to the location of resources used in the journaled research, the loss of which may make the research less transparent. We examined whether the hyperlinked resources were accessible by writing a set of scripts, accessible at https://github.com/teuben/ascl-tools, to automate the gathering and testing of hyperlinks in research papers.

The script *process_pdfs.py* extracts every hyperlink from the PDFs of every research paper published in *A&A* in 2015 (excluding Letters) using pyPdf v1.13 (Fenniak 2014), other than those hyperlinks that point to eight common and reliable sites (aanda.org, linker.aanda.org, arxiv.org, ascl.net, dx.doi.org, dexter.edpsciences.org, adsabs.harvard.edu, ui.adsabs.harvard,edu), those that use the doi:, email:, or mailto: protocols, and those that contain an at (@) sign, and stores them in the SQLite database links.sqlite. (All hyperlinks containing an at sign in our set of papers were email addresses with no protocol specified.)

The script *check_links.py* tests each HTTP(S) link and stores the response code and message returned by the server, or, in the case of an error that prohibits the sending of a request to the link (e.g. a malformed URL), a custom error code represented by a negative number. Links with no protocol specified are assumed to be HTTP.

[13] see VanderPlas (2017) for a discussion of various implementations

After the first run of this script, we edited the database to correct those malformed links that could be corrected (e.g. http:\\ instead of http://) and remove those that could not be (e.g. a spurious 'a'). We debated whether we should be doing even this minimum editing, but decided for it, as it is reasonable to assume a reader would know how to correct these small errors.

We ran this script four times on four dates from two different locations on our database of hyperlinks. The database generated is available at links.sqlite in the aforementioned GitHub repository. We checked FTP links by hand four times by entering them manually into our usual web browsers, Opera for one of us and Firefox for the other.

Our script *link_data.py* gathers the statistics from the database that we use in this paper: how many links consistently worked or failed to work, how many links inconsistently worked or failed for inconsistent reasons, and a count of links that consistently returned each attested error code.

4. RESULTS

4.1. *Software in articles*

The 166 papers we examined generated 715 instances of definite or possible software use; of these, we have examined 418 (58.5%) of the entries, those with codes identified by name, URL, ASCL ID, or otherwise easily determined to be software, to learn how available these packages or routines are. The other 41.5% of the entries will be examined further at a later date for possible additional research.

We categorized the 418 records examined into one of our ten categories; the number and percentage of entries in each category are shown in Table 4.

The 418 software usage instances are for 285 unique codes; those used in four or more papers are shown in Table 5; a frequency table for all 285 unique codes is provided as Appendix A. Of the 285 codes used in the 166 papers, 229 were used in only one paper if we count use of the Lomb-Scargle periodigram algorithm separately, which we do because of its different implementations.

The number of unique codes used in the examined papers and the categories we assigned them to are in Table 6.

Most software packages were used in only one paper in the sample. The question arose as to whether this software was "use once and never again." The answer is overwhelmingly no. We did full text searches (using ADS's Bumblebee) on 20% of these code names and found previous or subsequent use for almost every one of them. The median of the results from ADS was sixty-five articles. Though it is likely some results retrieved by doing these full text searches may not truly indicate use of the software, we are confident these results do show that nearly all of the codes used once in our sample set have been used again.

As mentioned in the Methods section, we emailed some code authors when we did not find a download site for a package through other methods, sending 46 emails

Table 4. Software availability categorization summary

Category	Number of entries	Percentage of entries
A	262	62.7%
B	26	6.2%
C	4	1.0%
N	70	16.7%
W	21	5.0%
GS	5	1.2%
GH	16	3.8%
P1	6	1.4%
P2	6	1.4%
O	2	0.5%
Total	418	100%

Table 5. Unique code category and frequency (4 or greater)

Code name	Category	Frequency
IRAF	A	31
SExtractor	A	10
HIPE	B	7
GILDAS	A	5
BGM	W	4
CASA	A	4
CIAO	A	4
DAOPHOT	A	4
DAOSPEC	GH	4
MOOG	A	4
MPFIT	A	4
SAS: Science Analysis Software	GS	4
XSPEC	A	4

Table 6. Unique code category summary

Category	Number of unique codes	Percentage of unique codes in category
A	162	56.8%
B	14	4.9%
C	4	1.4%
N	63	22.1%
W	16	5.6%
GS	2	0.7%
GH	10	3.5%
P1	6	2.1%
P2	6	2.1%
O	2	0.7%
Total number of unique codes	285	100.0%

representing 49 different codes. Of the 19 replies we received (by 30 September 2017), 12 expressed an interest in or intent to release their software eventually, with the timeline to making the code public ranging from a few weeks to a few years. One query provided source code to be housed on the ASCL, thus making it available, and we received URLs for software download sites for three codes that our searching had been unable to find. As a result of our email campaign, four codes moved from the *N* category ("we cannot find it") to the *A* category ("source code is available for download without restriction or impediment"), though one would not have been available otherwise.

In a quick evaluation done in 2012, our response rate to similar emails to software authors was 33%, divided among those who let us know the software we were looking for was not available (20%) and those that yielded source code for download (13%), as mentioned in Shamir et al. (2013). We are very pleased to have a response rate of 41% to our recent inquiries, though our true goal is code yield – how many codes of those we asked for were or became accessible; this effort gave us a yield of 8.7% (4 codes/46 emails).

4.2. *Link checking*

Our script extracted 2,591 hyperlinks from the PDFs of every research paper published in *A&A* in 2015, excluding those for the eight domains mentioned previously. After removing malformed links, 2,558 hyperlinks remained. Of these, 30 used the FTP protocol; the rest used HTTP(S). We checked the FTP links by hand and found

Table 7. FTP links accessibility status
Total links: 30

Status	Number of links
Always accessible	25
Intermittently accessible	1
Always inaccessible	4

that four were consistently unreachable, one was intermittently inaccessible, and the other 25 were consistently accessible (see Table 7).

We checked the HTTP(S) links with *check_links.py* four times, on two different computers in two different locations on four different dates in September and October 2017, to avoid counting links as broken that were only temporarily unavailable due to (for example) routine server maintenance. We found that 267 HTTP(S) links and four FTP links, or 10.6% of all links, were unreachable (did not return a 200 OK) in all runs of the script. See Table 8 for the error codes returned by the unreachable HTTP(S) links.

This statistic, however, differs slightly from the true number of unreachable links. Three links returned 3XX redirect codes, which were counted as unreachable but should not be assumed to be so (in fact, one was accessible), and ten links were deemed inaccessible due to SSL certificate errors, though manual checking showed that nine were accessible. On the other hand, some links that returned inconsistent results likely became inaccessible in the period between the first and last runs of the script. However, most consistently inaccessible links returned status codes such as 404 or -1 (our coding for inability to reach the site at all), which are not likely to be temporary conditions or actually accessible, and there were very few (65) inconsistently accessible links, so these factors have a negligible impact on the end result: if only the 244 HTTP(S) links that consistently returned the status codes -1, 403, and 404 are truly inaccessible (a clear underestimate), 9.5% of links are inaccessible, and if all 338 links that we failed to access even once are truly inaccessible (a clear overestimate), 13.2% are inaccessible.

5. DISCUSSION AND CONCLUSION

As of 30 September 2017, the codes used in this body of research – in our 166 articles – that are immediately available for a reader of these articles are those in the *A* and the *GS* categories, 57.5% of the software identified in these papers. If we add codes from the *P1* category, for purchasable source codes as in *Numerical Recipes* (and indeed, all codes in this category came from one edition or another of that resource),

Table 8. Error codes for and number of links consistently unreachable [a]

Error code	Message	Number of links
-4	httplib.BadStatusLine	1
-3	socket.error	2
-2	ssl.CertificateError	10
-1	urllib2.URLError (lookup failed)	74
301	Moved permanently (redirect)	2
302	Found	1
401	Unauthorized	2
403	Forbidden	25
404	Not found	145
500	Internal server error	4
502	Bad gateway	0
503	Service unavailable	1
504	Gateway timeout	0

[a]Negative error codes denote our coding for links that did not supply an error message

a volume that is commonly available to researchers, the figure rises to 59.6%, 170 of the 285 unique codes from our examined articles.

We were unable to find source code online, or it has a hard gate, is available only as an executable or only to collaborators, must be purchased, is a web-accessible black box, or is otherwise not immediately available to the researcher who might like to see how something was calculated, for 115 codes, 40.4% of the codes used in the research we looked at.

What does this mean for the repeatability of the research shared in these articles? And how transparent is any individual paper?

Certainly the relatively broad use of IRAF in research – appearing in 31 papers (as components or in whole), 18.7% of our sample – and other popular open access packages has a positive impact on the transparency of astrophysics research. We would expect to see AstroPy (AstroPy Collaboration, 2013), for example, occupy a similar spot in future research. Such packages improve the discipline in a number of ways in addition to providing transparency, including improving efficiency, as others do not have to write software for the functionality these packages provide.

Indeed, when we look not at the statistics for unique codes, but instead at the use of software spread throughout the body of these 166 papers, the availability of source code increases in comparison. Taking the use of category *A*, *GS*, and *P1* codes across all the papers, 65.3% of the software used in the papers had source code available. The use of IRAF and other open source packages such as SExtractor (Bertin & Arnouts 1996), GILDAS (GILDAS Team 2013), and CASA (McMullin et al. 2007) in multiple papers weights the overall "availability index," we might say, favorably upward.

There is no guarantee that the code we found was the version used in the article(s) in which we found the software, as we did not look for any specific version. Our ability to find the software now does not mean the software was available in 2015 when these articles were published, nor does our inability to find source code today in 2017 mean that it was unavailable when the research was published. This study is a snapshot in time; we do not make any claims about our numbers standing for software availability for astrophysics research overall.

At the time we conducted our research (August-October 2017), the articles we examined had been published only two years before. Assuming the "best case" for the hyperlinks we tested, 9.7% of the links were inaccessible when we tested them. We have made the links SQLite database available on the aforementioned GitHub repository, and have also made it available in Figshare. We plan to test these links periodically to see what the rate of loss is over time, and we plan to build out the database in the future to get a more complete picture of link persistence.

It would be interesting to categorize the links into categories by their destinations, whether for software, data, websites, or other resources, and compare link accessibility not only through time but also by destination category, and we may do this in subsequent research.

Our preliminary efforts may be flawed in various ways; we may have missed information about code use in the articles we looked at and included information about data in our larger dataset of 715 entries. In fact, we know this latter happened, but this is mitigated in this study by our excluding records that did not clearly identify software usage. In particular, we might be under-reporting commercial code use slightly, given the first author's proclivity to focus on researcher-written software. We may have miscategorized one or more software packages, either through an inability to find a site for the source code, lack of knowledge or experience with one or more packages and components, or through misunderstanding text about or reference for a code. We may have been overly strict about some software, such as the Lomb-Scargle periodigram algorithm, in wanting to know exactly which implementation was used. In not scoring records that did not have easily identifiable software, we almost certainly left software to examine on the table. It is very likely we did not find all the URLs in the articles we examined, as we pulled back only those that were coded as active hyperlinks, and we may have errors in one or more of our scripts. We have made all of our software, the records used for this paper, and other artifacts of our research available so that others can vet the information we provide here.

Still, this is a beginning, and we are heartened to see more software has source code available than we initially thought and thank the software and article authors for making that so. We will make use of what we have learned in conducting this study and apply it to the next.

6. ACKNOWLEDGMENTS

This research has made use of NASA's Astrophysics Data System; indeed, this research would have been nearly impossible without it. Our thanks to Kimberly DuPrie, Lior Shamir, Bruce Berriman, and Robert Nemiroff for reading articles for software usage mentions, to Lior Shamir, Robert Nemiroff, Bruce Berriman, Tracy Huard, James Howison, and Robert Haines for helpful conversations, and to James Howison for continuing email correspondence about this project. The anonymous referee provided encouragement and a helpful report and we are grateful for this input and thank the referee for it. We thank the authors of the papers we examined, the scientists who wrote the software that enabled that research, and *A&A* for making its articles open access.

REFERENCES

Ahalt, S., Carsey, T., Couch, A., et al. 2015, NSF Workshop on Supporting Scientific Discovery Through Norms and Practices for Software and Data Citation and Attribution, Tech. rep., USA

Allen, A., Berriman, B., Brunner, R., et al. 2013, in Astronomical Society of the Pacific Conference Series, Vol. 475, Astronomical Data Analysis Software and Systems XXII, ed. D. N. Friedel, 383

Baker, M. 2016, Nature, 533, 452

Barnes, N. 2010, Nature, 467, doi:10.1038/467753a. `http://www.nature.com/news/2010/101013/full/467753a.html`

Bertin, E., & Arnouts, S. 1996, A&AS, 117, 393

Bonanno, A., & Urpin, V. 2015, A&A, 574, A63

Brown, D. A., Brady, P. R., Dietz, A., et al. 2007, A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis, ed. I. J. Taylor, E. Deelman, D. B. Gannon, & M. Shields (London: Springer London), 39–59. `https://doi.org/10.1007/978-1-84628-757-2_4`

Campana, R. 2017, pyLCSIM: X-ray lightcurves simulator, Astrophysics Source Code Library, , , ascl:1708.016

Collberg, C., & Proebsting, T. A. 2016, Commun. ACM, 59, 62. `http://doi.acm.org/10.1145/2812803`

Fenniak, M. 2014, pyPdf: PDF Toolkit, Python Package Index, Python Package Index. `https://pypi.python.org/pypi/pyPdf`

GILDAS Team. 2013, GILDAS: Grenoble Image and Line Data Analysis Software, Astrophysics Source Code Library, , , ascl:1305.010

Goble, C., Howison, J., Kirchner, C., Nierstrasz, O., & Vinju, J. J. 2016, Dagstuhl Reports, 6, 62. `http://drops.dagstuhl.de/opus/volltexte/2016/6755`

Hettrick, S., Antonioletti, M., Carr, L., et al. 2014, UK Research Software Survey 2014, , , doi:10.5281/zenodo.14809. `https://doi.org/10.5281/zenodo.14809`

Howison, J., & Bullard, J. 2016, Journal of the Association for Information Science and Technology, 67, 2137. `http://dx.doi.org/10.1002/asi.23538`

Howison, J., Deelman, E., McLennan, M. J., Ferreira da Silva, R., & Herbsleb, J. D. 2015, Research Evaluation, 24, 454. +http://dx.doi.org/10.1093/reseval/rvv014

Ince, D. C., Hatton, L., & Graham-Cumming, J. 2012, Nature, 482, 485

Joppa, L. N., McInerny, G., Harper, R., et al. 2013, Science, 340, 814

Kalamkar, M., Casella, P., Uttley, P., et al. 2016, MNRAS, 460, 3284

Katz, D. S., Choi, S.-C. T., Lapp, H., et al. 2014, ArXiv e-prints, arXiv:1404.7414

Limare, N. 2012, in ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, Providence, United States, http://icerm.brown.edu/tw12-5-rcem. https://hal.archives-ouvertes.fr/hal-00783292

Lomb, N. R. 1976, Ap&SS, 39, 447

Ludaescher, B., Chard, K., Gaffney, N., et al. 2016, ArXiv e-prints, arXiv:1610.09958

Marwick, B. 2015, How computers broke science &ndash; and what we can do to fix it, , . http://theconversation.com/how-computers-broke-science-and-what-we-can-do-to-fix-it-4993

McMullin, J. P., Waters, B., Schiebel, D., Young, W., & Golap, K. 2007, in Astronomical Society of the Pacific Conference Series, Vol. 376, Astronomical Data Analysis Software and Systems XVI, ed. R. A. Shaw, F. Hill, & D. J. Bell, 127

Momcheva, I., & Tollerud, E. 2015, ArXiv e-prints, arXiv:1507.03989

Morin, A., Urban, J., Adams, P. D., et al. 2012, Science, 336, 159

NASA. 2014, NASA Plan for Increasing Access to the Results of Scientific Research, Tech. rep. http://www.nasa.gov/sites/default/files/atoms/files/206985_2015_nasa_plan-for-web.pdf

NASA High Energy Astrophysics Science Archive Research Center (Heasarc). 2014, HEAsoft: Unified Release of FTOOLS and XANADU, Astrophysics Source Code Library, , , ascl:1408.004

Press, W. H., Flannery, B. P., & Teukolsky, S. A. 1986, Numerical recipes. The art of scientific computing

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992a, Numerical recipes in C. The art of scientific computing

—. 1992b, Numerical recipes in FORTRAN. The art of scientific computing

—. 2002, Numerical recipes: the art of scientific computing

Santos-Sanz, P., Ortiz, J. L., Morales, N., et al. 2015, A&A, 575, A52

Scargle, J. D. 1982, ApJ, 263, 835

Shamir, L., Wallin, J. F., Allen, A., et al. 2013, Astronomy and Computing, 1, 54

Smith, A. M., Katz, D. S., Niemeyer, K. D., & FORCE11 Software Citation Working Group. 2016, PeerJ Computer Science, 2:e86, doi:10.7717/peerj-cs.86

Smith, A. M., E Niemeyer, K., Katz, D. S., et al. 2017, ArXiv e-prints, arXiv:1707.02264

Soito, L., & Hwang, L. J. 2016, International Journal of Digital Curation, 11, doi:10.2218/ijdc.v11i2.390. https://doi.org/10.2218%2Fijdc.v11i2.390

Stetson, P. B. 1987, PASP, 99, 191

Stodden, V. 2010, The Scientific Method in Practice: Reproducibility in the Computational Sciences, Tech. Rep. MIT Sloan Research Paper No. 4773-10, doi:10.2139/ssrn.1550193

Stodden, V., McNutt, M., Bailey, D. H., et al. 2016, Science, 354, 1240

Taddia, F., Sollerman, J., Fremling, C., et al. 2015, A&A, 580, A131

Teuben, P., Allen, A., Berriman, B., et al. 2014, in Astronomical Society of the Pacific Conference Series, Vol. 485, Astronomical Data Analysis Software and Systems XXIII, ed. N. Manset & P. Forshay, 3

Tody, D. 1986, in Proc. SPIE, Vol. 627, Instrumentation in astronomy VI, ed. D. L. Crawford, 733

Tody, D. 1993, in Astronomical Society of the Pacific Conference Series, Vol. 52, Astronomical Data Analysis Software and Systems II, ed. R. J. Hanisch, R. J. V. Brissenden, & J. Barnes, 173

Tubiana, C., Snodgrass, C., Bertini, I., et al. 2015, A&A, 573, A62

VanderPlas, J. T. 2017, ArXiv e-prints, arXiv:1703.09824

Vitek, J., & Kalibera, T. 2011, in Proceedings of the Ninth ACM International Conference on Embedded Software, EMSOFT '11 (New York, NY, USA: ACM), 33–38. `http://doi.acm.org/10.1145/2038642.2038650`

Vrard, M., Mosser, B., Barban, C., et al. 2015, A&A, 579, A84

Weiner, B., Blanton, M. R., Coil, A. L., et al. 2009, in ArXiv Astrophysics e-prints, Vol. 2010, astro2010: The Astronomy and Astrophysics Decadal Survey, 61P

APPENDIX

A. CODE FREQUENCY AND CATEGORY

Table 9. Code frequency and category

Code name/text	Frequency	Category
IRAF	31	A
SExtractor	10	A
HIPE	7	B
GILDAS	5	A
BGM: Besançon Galaxy model	4	W
CASA	4	A
CIAO	4	A
DAOPHOT	4	A
DAOSPEC	4	GH
MOOG	4	A
MPFIT	4	A
SAS: Science Analysis Software	4	GS
XSPEC	4	A
ARES	3	A
BEC	3	N
ESO GIRAFFE pipeline	3	A
GARSTEC	3	GH
Gasgano	3	B
HEASoft	3	A
Libre-Esprit	3	N
MARCS	3	A
PHOEBE	3	A
pPXF	3	A
Reflex	3	B
Scanamorphos	3	A
SPICE	3	A
TOPCAT	3	A

Note—Table 9 is not published in its entirety here; it will be available in a machine-readable format when the article is published. A portion is shown here for guidance regarding its form and content.

B. SAMPLE EMAIL TO SOFTWARE AUTHORS

```
Subject line: [code name]

Dear Dr. [name],

I came across [code name] when looking for astrophysics codes. I would
like to add this code to the Astrophysics Source Code Library (ASCL,
ascl.net). The ASCL is indexed by ADS and Web of Science; this
indexing provides useful research software with greater visibility.

I could not find a link to the source code, however. Is the code
available for download, and if so, what is the URL? If there is no
download site, the ASCL can house an archive file of the code if you
prefer.

The ASCL seeks to increase the transparency of astrophysics research
by making codes discoverable for examination and we are always
looking for codes which have been used in refereed research. If you
have additional codes that should be listed, I am interested in
knowing about them.

Thank you for your attention.

Clear skies,

Alice Allen
Editor, ASCL
```

Table 10. IDs and links to PDFs of examined articles

Article ID	Link to PDF
aa23704-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa23704-14.pdf
aa23996-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa23996-14.pdf
aa24235-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24235-14.pdf
aa24356-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24356-14.pdf
aa24393-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24393-14.pdf
aa24472-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24472-14.pdf
aa24589-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24589-14.pdf
aa24675-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24675-14.pdf
aa24691-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24691-14.pdf
aa24735-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24735-14.pdf
aa24967-14	https://www.aanda.org/articles/aa/pdf/2015/01/aa24967-14.pdf

Note—Table 10 is not published in its entirety here; it will be available in a machine-readable format when the article is published. A portion is shown here for guidance regarding its form and content.

C. IDS AND LINKS TO PDFS OF EXAMINED ARTICLES