Software, Free

Brandon LeBeau[1]

[1] University of Iowa

Software, Free

Statistical software is an enabling component of conducting educational research. In particular, it makes statistical routines such as linear regression, data visualization, data manipulation, or other data related tasks easier for users to perform. There are a variety of statistical software that users can choose from, these would include general purpose software such as SPSS (IBM Corp, 2019), Stata (StataCorp, 2019), SAS (SAS Institute, 2017), R (R Core Team, 2021), Python (Rossum, 1995; Van Rossum & Drake, 2009), and Julia (Bezanson et al., 2017). Furthermore, thematic coding of qualitative studies are also commonly used to aid in the coding element using software such as NVivo (QSR International Pty Ltd., 2020). The focus on this entry will be on quantitative software as these are more numerous, but the principles can be applied to qualitative software (or more generally any software) as well.

In addition to the general purpose statistical software, there are numerous specialized pieces of software that are designed to perform one type analysis or methodology. Some common specialty statistical software would include, MPlus for latent variable modeling (Muthén & Muthén, 1998-2017), flexMirt for item response theory modeling (Cai, 2013), G*Power3 for estimating statistical power from common simple statistical designs (Faul et al., 2007), or optimal design for estimating power for experimental designs (Raudenbush, 2011).

Within the general purpose and specialty statistical software groups, the software generally falls into two broad categories, those software that are freely available for users to use compared to those that require a fee to use. For example, SPSS, SAS, or Stata all require users to purchase a license to use the statistical software. General purpose statistical software such as R, Python, Julia or specialty software such as G*Power3 or optimal design are free software. Free software is usually available to download and install from a website and users can use the software without limitations.

This encyclopedia entry will first focus on differentiating between free and paid

software. free software and discuss differences within this broad category. Prior to will be articulated, including the difference between the two. Strengths and weaknesses of different frameworks within this category will also be explored in more detail in this entry.

Paid vs Free vs open source software

The distinction between free and paid software is whether the software license has a cost associated with it. In particular, if the software requires a fee to be paid then it is paid software. If the usage of the software does not require a fee, then it would be free. Within the free software paradigm, the focus of this entry, there is a more subtle distinction that has important implications.

The distinction between free and open source software is important and is related to whether the source code is available or not. A common definition for open source software is that the source code is freely available, but also that the source code can be inspected, modified, or enhanced by anyone (*What Is Open Source?*, n.d.). Free software, however, could be freely available to download, install, and use, but the source code may not be able to be seen. If the source code is unable to be inspected, this is often referred to as proprietary or closed source software. In these instances, the development team is the only one who can legally access, modify, enhance, or distributed the source code.

Examples of free or open source software may help to show differences in these approaches. Open source statistical software would include R, Julia, Python, and the packages associated with them. In contrast, statistical software such as G*Power3 or Optimal Design are free software, but the source code associated with them to implement the features is not available outside the development team. Furthermore, as has been implied throughout, it is possible to have free software but not be open source, but the inherent nature of open source software is such that it is always free. In contrast, software that is paid software would always need to be closed source such that the source code was not

openly available to its users. Free software could be either open source or closed source depending on the developer teams preferences.

What is source code?

In the previous section, the term source code was used, but what exactly would represent source code? Source code is the code that developers use to implement specific features within a piece of software and in which many users of the software do not see (*What Is Open Source?*, n.d.). If the software is a command line or code based software (e.g., R or Python), the source code would not be the code that users use to perform a specific task. Rather, the source code would be the code that a developer wrote to implement that routine.

A specific example may be helpful. Linear regression is a common analysis that is performed in educational research. This command in R is `lm()` and the user interacts with this code by first typing the `lm()` command in a script or within the terminal in R. The source code for the linear regression implementation in R can be viewed within an R session by typing `lm` without any parentheses around it. This source code processes the users specifications via function arguments to conduct the desired analysis.

R is an open-source language, therefore the source code is directly viewed by anyone who wants to see the source code. However, there is source code underlying any software and this code is ran when the user requests a specific routine. For example, within SPSS, a linear regression analysis can be performed in at least two ways by an end user. One is through the graphical user interface and menu structure (e.g., Analyze > Regression > Linear) or through the SPSS syntax, `REGRESSION`. Whichever method is used, there is source code behind the scenes that is implementing this analysis, however the primary difference between R and SPSS is that SPSS is a paid, closed-source software. This would mean that a user has to purchase a software license to use the routine and also since it is closed-source, the user is unable to see the specific source code used to run the routine behind the scenes.

Strengths of open source software

The strengths of open source software will be explored explicitly in this section. The benefits of the open source software stem from the ability to directly view, modify, enhance, study, or distribute the source code. Having access to the source code can improve collaboration of users with the developer. This can lead to improvements to the software, including better documentation, enhancing usability, or even fix errors to the implementation of the software routines. The fixing of errors in the implementation of a software routine is an important distinction for open source software as this would not directly be possible within a closed source software project. Instead, the user would need to contact the developer directly to try to explain the problem and hopefully get a fix in the next release of the software.

Furthermore, open source software can be extended to add additional functionality on top of an existing open source software project. As the original open source project is freely distributed and the source code is accessible, there is no need to re-implement the functionality of the original software project. The new functionality can be added directly, which from a developer standpoint can increase productivity to focus on the new elements rather than re-implement a problem that has already has a solution.

Weakness of open source software

The primary weakness of open source software is that the software is distributed and licensed as-is without any warranty or support. From the user perspective, this does put more burden to learn the software themselves as the developer makes no commitment to answer usability questions. It should be noted here, that the support and warranty for free closed source software is also most likely distributed and licensed as-is. Therefore, there is likely not a large difference in the support and warranty for the source code of free and open source compared to free and closed source software.

Another potential weakness of open source software could include the ability of

developers to fix errors in a timely manner. As discussed above, the software is often distributed as-is, therefore the developer is under no obligations to provide a fix for potential errors. The developers may also not be trained software developers which could make the software less flexible or more prone to errors. These need to be watched when using open source software. Similar to the note above, these issues may also be found in free closed source software, however in this scenario the source code could not be viewed to evaluate if the source code does have an error or not.

Some open source software may provide support as an add-on service that users could pay for, but the software is still considered open source (i.e., the source code is made accessible). This idea is similar to support provided from paid software where users often are able to opt in to receive additional support on top of paying for the software. Some paid software may also provide free support when you purchase the software, but this is not necessarily required as part of the paid software.

Software Licenses

Much of the distinction between paid, free, or open source software comes down to the software license that is distributed with the software. There are largely two buckets that licenses fall into, those that are free and open vs those that are proprietary. Within these ecosystems, there is quite a bit of variety to the software licenses and many slightly different versions of licenses. This entry will not be able to give a deep dive of all the software license options, but a few specific and more common software licenses will be discussed and compared.

The primary differences between the two large groups of software licenses comes down to what actions are typically allowed with respect to the source code. Software distributed under a proprietary license typically are closed source and do not allow the source code to be viewed, distributed, or modified. This is in contrast to the free and open licenses which

require the software to provide the source code.

Free and open licenses

There is quite a variety to free and open licenses, but all of these licenses require that the source code be released, viewed, modified, and distributed. There are numerous subtle differences across the variety of licenses available, but these can be broken down into two broad categories. One group are referred to as permissive licenses (e.g., MIT, Apache, BSD, or MPL licenses) and the second group is referred to as copyleft licenses (e.g., GPL or AGPL). A more thorough list of free and open source licenses can be viewed at the Free Software Foundation website (https://www.fsf.org/licensing/) or the Open Source Initiative (https://opensource.org/licenses).

Permissive Licenses

Permissive licenses are open source licenses that give broad permissions to use the software. In particular, there are no limitations on using, modifying, or redistributing the software. The notable difference with permissive licenses to copyleft licenses is that the permissive licenses place very few, if any, restrictions on how the software can be redistributed. For example, software with a permissive license that is free and open source can be used within closed source software or released under a proprietary license. The hallmark of permissive licenses is that, in general, the software can be used however the user desires with very few restrictions.

There are many instances of permissive licenses used in open source projects, for example Python uses a permissive license that uses their own Python Software Foundation license and also licenses some with the BSD license. Julia also uses the MIT license for their software project. These permissive licenses give broad use for the software to those who wish to use, build, enhance, view, or distribute part of all of the source code.

Copyleft Licenses

Copyleft licenses are similar to that of permissive licenses, however these licenses have one important distinction. If a copyleft license is used, such as the GNU General Public License (GPL), there are greater restrictions on how the software can be distributed. In particular, software with a copyleft license must remain open source, free, and with the same license. For instance, if software is built that uses functionality from a GPL licensed software, the new software must also be open source with the source code released. Furthermore, implied within the last example, software that has a copyleft license can not be used inside software that is uses a proprietary license or require a fee to use the software. The copyleft licenses are also a type of "share-alike" license where the license is designed to restrict new software that builds off of software with a copyleft license to be released with the same or similar type of license.

Similar to the permissive licenses, copyleft licenses are used frequently within open source work. For example, R uses the GPL v2 license for its source code.

Proprietary licenses

Proprietary licenses limit the usage of the software and source code. Users are often asked to agree to an end-user license agreement to use the software. These agreements typically limit who can use the software, restrict the sharing of the software, or releasing of any source code. Furthermore, ownership of the software is retained by the publisher. Examples of proprietary software within educational research would be, SAS, SPSS, MPlus, Stata, or NVivo.

Software such as Optimal Design or G*Power3 would follow a proprietary free license. In this situation, the source code is not shared, but the software is freely downloaded and used by end-users. Users do agree to not distribute or modify the source code however, therefore these software would fall best under a proprietary license even though they are free

to download and use.

Package ecosystems of open source software

The open source licenses of R, Python, and Julia have allowed software developers, hobbyists, and others to build additional functionality on top of each software. This is made possible because of the open source licenses that these software have chosen to adhere to. Most, if not all, of the software packages that have been developed to enhance R, Python, or Julia are also open source with permissive or copyleft licenses.

All of these package ecosystems continue to evolve over time. As of writing this entry, there are over 17,000 software packages that were written to extend functionality of R, over 200,000 Python packages (not all of these would be relevant for educational research however), and about 4,000 Julia packages. Much of the functionality that has been added to these general purpose languages would not have been developed for the platform without the distributed effort of the many developers that have contributed their time and expertise. This is an example of the open source community presenting additional benefits over the closed source development process.

Conclusions

Free software is a diverse and thriving area of software development. There are differences within free software that are important for the use of those software in the future. Much of these differences stem from the license that the software is released with, whether the software is a free and open source license or a proprietary license. Both of these types of licenses have there strengths and weaknesses, but the difference largely comes down to whether the source code is released, able to be viewed, or can be redistributed. If it can, then the software uses a free and open license, if not, then the software uses a proprietary license.

Using free and open licenses have many benefits, for instance allowing any user to view, modify, incorporate fixes, or build new functionality on top of existing source code.

This has directly led to the creation of package ecosystems on top of the primary open source package for R, Python, and Julia to add specialty functionality. Standalone free software that is not open source will always have a place in the software ecosystem, but the proliferation of open source software will likely continue as the benefits, in the opinion of the author, far outweigh any potential weaknesses.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98.

Cai, L. (2013). flexMIRT version 3.51: Flexible multilevel multidimensional item analysis and test scoring [computer software]. *Chapel Hill, NC: Vector Psychometric Group*.

Faul, F., Erdfelder, E., Lang, A.-G., & Buchner, A. (2007). G* power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods*, *39*(2), 175–191.

IBM Corp. (2019). *IBM SPSS Statistics*.

Muthén, L., & Muthén, B. (1998-2017). *Mplus users's guide. Eighth edition.*

QSR International Pty Ltd. (2020). *NVivo*.

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Raudenbush, S. W. et al. (2011). *Optimal design software for multi-level and longitudinal research (version 3.01) [computer software]*.

Rossum, G. van. (1995). *Python tutorial* (CS-R9526). Centrum voor Wiskunde en Informatica (CWI).

SAS Institute. (2017). *Base SAS 9.4 procedures guide: Statistical procedures*. SAS Institute.

StataCorp. (2019). *Stata statistical software: Release 16*.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

What is open source? (n.d.). https://opensource.com/resources/what-open-source.