OPEN SOURCE AND MORTGAGE DATA MODELING

A CTO'S GUIDE TO OPEN SOURCE

Release 1.0
October 2017

TABLE OF CONTENTS

INTRODUCTION	2
WHAT IS OPEN SOURCE SOFTWARE?	3
ADVANTAGES OF OPEN SOURCE PROGRAMS	4
RISKS OF OPEN SOURCE PROGRAMS	7
OPEN SOURCE SOFTWARE FOR MORTGAGE DATA ANALYSIS	11
USING OPEN SOURCE DATA MODELING TOOLS	12
REFERENCES	14
ABOUT RISKSPAN	15

Introduction

The growing relevance of open source software has changed the way large organizations approach their software solutions. While open source software was at one point rare in an enterprise's system, it's now the norm. A survey conducted by Black Duck Software revealed that fewer than 3% of companies don't rely on open source at all.[1] Even the most conservative organizations are hopping on board the open source trend.

In a blog post from June 2016, TechCrunch writes:

> *"Open software has already rooted itself deep within today's Fortune 500, with many contributing back to the projects they adopt. We're not just talking stalwarts like Google and Facebook; big companies like Walmart, GE, Merck, Goldman Sachs — even the federal government — are fleeing the safety of established tech vendors for the promises of greater control and capability with open software. These are real customers with real budgets demanding a new model of software."*[2]

The expected benefits of open source software are alluring all types of institutions, from small businesses, to technology giants, to governments. This shift away from proprietary software in favor of open source is streamlining operations. As more companies make the switch, those who don't will fall behind the times and likely be at a serious competitive disadvantage.

Still, financial institutions have traditionally been slow to adopt the latest data and technology innovations due to the strict regulatory and risk-averse nature of the industry, and open source has been no exception. As open source becomes more mainstream, however, many of our clients have come to us with questions regarding its viability within the mortgage industry.

The short answer is simple: open source has a lot of potential for the financial services and mortgage industries, particularly for data modeling and data analysis. Within our own organization, we frequently use open source data modeling tools for our proprietary models as well as models built for clients. While a degree of risk is inherent, prudent steps can be taken to mitigate them and profit from the many worthwhile benefits of open source.

This paper will explore the various advantages and risks inherent in using open source software, as well as ways to mitigate negative impacts. But first, let's cover some basics.

[1] https://techcrunch.com/2016/06/19/the-next-wave-in-software-is-open-adoption-software/
[2] https://techcrunch.com/2016/06/19/the-next-wave-in-software-is-open-adoption-software/

What Is Open Source Software?

Software has conventionally been considered open source when the original code is made publicly available so that anyone can edit, enhance, or modify it freely. This original concept has recently been expanded to incorporate a larger movement built on values of collaboration, transparency, and community.

In contrast, proprietary software refers to software for which the source code is only accessible to those who created it. Thus, only the original authors have control over any updates or modifications. Outside players are barred from even viewing the code to protect the owners from copying and theft. To use proprietary software, users agree to a licensing agreement and typically pay a fee. The agreement legally binds the user to the owners' terms and prevents the user from any actions the owners have not expressly permitted.

Open source software, on the other hand, gives any user free rein to view, copy, or modify it. The idea is to foster a community built on collaboration, allowing users to learn from each other and build on each other's work. Like with proprietary software, open source users must still agree to a licensing agreement, but the terms differ significantly from those of a proprietary license.[3]

Mature institutions often have employees, systems, and proprietary models entrenched in closed source platforms. For example, SAS Analytics is a popular provider of proprietary data analysis and statistical software for enterprise data operations among financial institutions. But several core computations SAS performs can also be carried out using open source data modeling tools, such as Python and R. The data wrangling and statistical calculations are often fungible and, given the proper resources, will yield the same result across platforms.

Open source is not always a viable replacement for proprietary software, however. Factors such as cost, security, control, and flexibility must all be taken into consideration. The challenge for institutions is picking the right mix of platforms to streamline software development. This involves weighing benefits and drawbacks.

[3] https://opensource.com/resources/what-open-source

Advantages of Open Source Programs

The Cost of Open Source Software

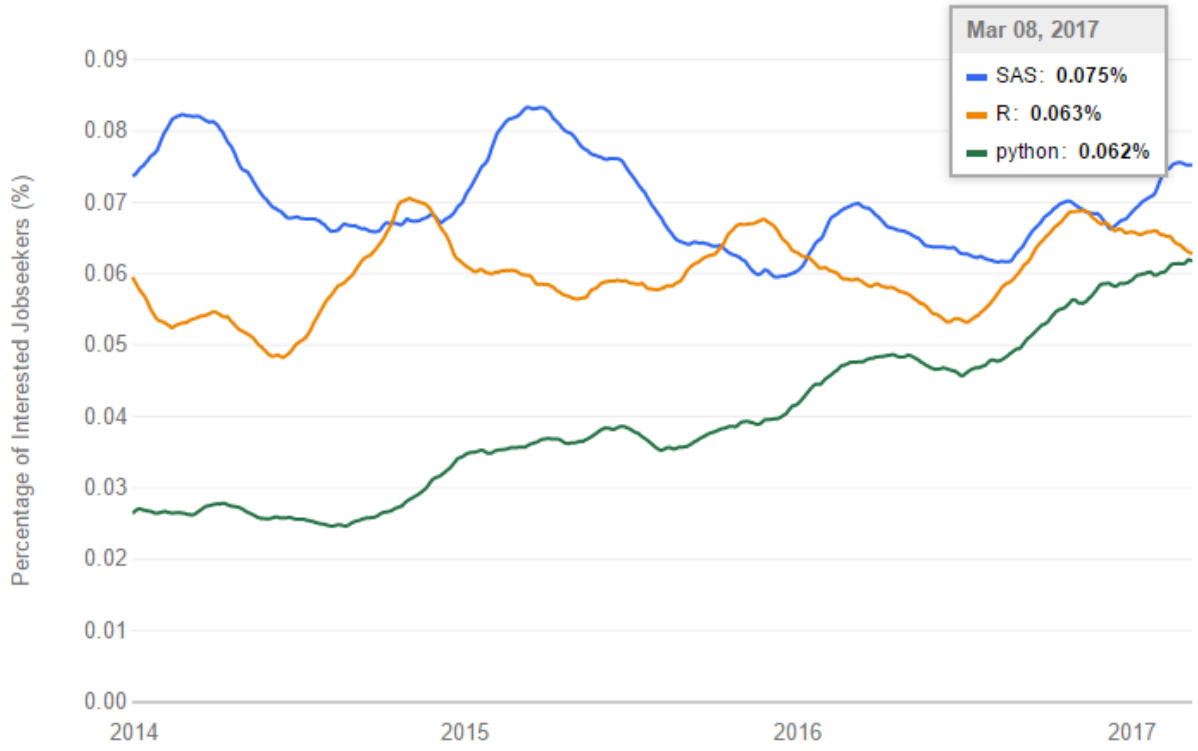
The low cost of open source software is an obvious advantage. Compared to the upfront cost of purchasing a proprietary software license, using open source programs seems like a no-brainer. Open source programs can be distributed freely (with some possible restrictions to copyrighted work) resulting in virtually no direct costs. However, indirect costs can be difficult to quantify. Downloading open source programs and installing the necessary packages is easy and adopting this process can expedite development and lower costs. On the other hand, a proprietary software license may bundle setup and maintenance fees for the operational capacity of daily use, the support needed to solve unexpected issues, and a guarantee of full implementation of the promised capabilities. Enterprise applications, while accompanied by a high price tag, provide ongoing and in-depth support of their products. The comparable cost of managing and servicing open source programs that often have no dedicated support is difficult to determine.

Open Source Talent Considerations

Another advantage of open source is that it attracts talent who are drawn to the idea of sharable and communitive code. Students and developers outside of large institutions are more likely to have experience with open source applications since access is widespread and easily available. Open source developers are free to experiment and innovate, gain experience, and create value outside of the conventional industry focus. This flexibility naturally leads to broader skilled inter-disciplinarians. Figure 1 shows a chart from Indeed's Job Trend Analytics tool which reflects strong growth in open source talent, especially Python developers.

From an organizational perspective, the pool of potential applicants with relevant programming experience widens significantly compared to the limited pool of developers with closed source experience. For example, one may be hard-pressed to find a new applicant with development experience in SAS since comparatively few have had the ability to work with the application. Key-person dependencies become increasingly problematic as the talent or knowledge of the proprietary software erodes down to a shrinking handful of developers.

Figure 1: Job Seekers Interests via Indeed



**Indeed searches millions of jobs from thousands of job sites. The jobseeker interest graph shows the percentage of jobseekers who have searched for SAS, R, and python jobs.*

Support and Collaboration

The collaborative nature of open source facilitates learning and adapting to new programming languages. While open source programs are usually not accompanied by the extensive documentation and user guides typical of proprietary software, the constant peer review from the contributions of other developers can be more valuable than a user guide. In this regard, adopters of open source may have the talent to learn, experiment with, and become knowledgeable in the software without formal training.

Still, the lack of support can pose a challenge. In some cases, the documentation accompanying open source packages and the paucity of usage examples in forums do not offer a full picture. For example, RiskSpan built a model in R that was driven by the available packages for data infrastructure – a precursor to performing statistical analysis – and their functionality. R does not have an active support solutions line and the probability of receiving a response from the author of the package is highly unlikely. This required RiskSpan to thoroughly vet packages.

Flexibility and Innovation

Another attractive feature of open source is its inherent flexibility. Python allows users to use different integrated development environments (IDEs) that have multiple different characteristics or functions, as compared to SAS Analytics, which only provides SAS EG or Base SAS. R makes possible web-based interfaces for server-based deployments. These functionalities grant more access to users at a lower cost. Thus, there can be more firm-wide development and participation in development. The ability to change the underlying structure of open source makes it possible to mold it per the organization's goals and improve efficiency.

Another advantage of open source is the sheer number of developers trying to improve the software by creating many functionalities not found in their closed source equivalent. For example, R and Python can usually perform many functions like those available in SAS, but also have many capabilities not found in SAS: downloading specific packages for industry specific tasks, scraping the internet for data, or web development (Python). These specialized packages are built by programmers seeking to address the inefficiencies of common problems. A proprietary software vendor does not have the expertise nor the incentive to build equivalent specialized packages since their product aims to be broad enough to suit uses across multiple industries.

RiskSpan uses open source data modeling tools and operating systems for data management, modeling, and enterprise applications. R and Python have proven to be particularly cost effective in modeling. R provides several packages that serve specialized techniques. These include an archive of packages devoted to estimating the statistical relationship among variables using an array of techniques, which cuts down on development time. The ease of searching for these packages, downloading them, and researching their use incurs nearly no cost.

Open source makes it possible for RiskSpan to expand on the tools available in the financial services space. For example, a leading cash flow analytics software firm that offers several proprietary solutions in modeling structured finance transactions lacks the full functionality RiskSpan was seeking. Looking to reduce licensing fees and gain flexibility in structuring deals, RiskSpan developed cashflow programs in Python for STACR, CAS, CIRT, and other consumer lending deals. The flexibility of Python allowed us to choose our own formatted cashflows and build different functionalities into the software. Python, unlike closed source applications, allowed us to focus on innovating ways to interact with the cash flow waterfall.

Risks of Open Source Programs

Deploying open source solutions also carries intrinsic challenges. While users may have a conceptual understanding of the task at hand, knowing which tools yield correct results, whether derived from open or closed source, is another dimension to consider. Different parameters may be set as default, new limitations may arise during development, or code structures may be entirely different. Various challenges may arise from translating a closed source program to an open source platform. Introducing open source requires new controls, requirements, and development methods.

Open Source Security Risks

Open source software is not inherently more or less prone to malicious code injections than proprietary software. It is true that anyone can push a code enhancement for a new version and it may be possible for the senior contributors to miss intentional malware. However, in these circumstances, open source has an advantage over proprietary, coined in 1999 by Eric S. Raymond as Linus's Law: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone."[4] It is unlikely that a deliberate security error goes unnoticed by the many pairs of eyes on each release.

However, security issues persist. Debian, a Unix-like computer operating system, was one of the first to be based on the Linux kernel. Like many systems, it utilizes OpenSSL, a software library that provides an open source implementation of the Secure Sockets Layer (SSL) protocol, commonly used by applications that require secure communications over a network.

In 2006, a snippet of code was removed from Debian's OpenSSL package after one of the contributors found that it caused runtime warnings generated by other packages. After the removal, the pseudorandom number generator (PRNG) generated SSL keys using only the process ID (in Linux, a number up to 32,768) to the exclusion of all other random data. Since a relatively small number of values was used, the keys created over a period of almost two years were too predictable to be used securely. Users became aware of the issue 20 months after the bug was introduced, leading to costly security resolutions for companies and individuals who relied on Debian's OpenSSL implementation.[5]

OpenSSL was again the subject of negative attention when a bug dubbed 'Heartbleed' was introduced to the code in 2012 and disclosed to the public in 2014. A fixed version of OpenSSL was released *on the same day* the issue was announced. More than a month after the release, however, 1.5% of the 800,000 most popular affected websites were still vulnerable to the security bug.[6]

[4] http://www.catb.org/esr/writings/cathedral-bazaar/
[5] https://www.theregister.co.uk/2008/05/21/massive_debian_openssl_hangover/
[6] http://www.theregister.co.uk/2014/05/20/heartbleed_still_prevalent/

The good news is that such vulnerabilities are documented in the Common Vulnerabilities and Exposures (CVE) system, and they are not so common. For Python 2.7, the popular version released in 2010, 15 vulnerabilities were recorded from 2010 to 2016, only one of which is considered 'High' severity, with a CVSS score of 7.5. jQuery, a JavaScript library that simplifies some components of web application development and the most common open source component identified in the latest Open Source Security and Risk Analysis (OSSRA) report, only has four known vulnerabilities from 2007 to 2017, none of which rank higher than a 'Medium'.

The CVE is just one tool available for improving the security profile of software applications but technologists must remain vigilant and abreast of known issues. Corporate IT governance frameworks should be continuously updated to keep up with the changing structure of the underlying technology itself.

Bad Code

Serious security vulnerabilities may not be a daily occurrence, but bad code can affect software at any time. *pandas*, a popular open source software library used in Python implementations for data manipulation and analysis, was first released in 2009. Since then, its contributors have identified over 10,000 issues, 1,933 of which are currently considered unresolved.[7] A company that relies on accurate output from a codebase that uses *pandas* needs to be vigilant not only in testing the code written by its in-house developers, but also in verifying that all outstanding known *pandas* issues are covered by workarounds and the rest of the functionality is sound. Developers and testers who are not intimately familiar with the *pandas* source code must devise creative testing tools to ensure complete integrity of applications that rely on it.

In addition to implementing security testing, IT controls must include a clear framework for testing both in-house and open source components of all applications, especially high-impact programs.

Redundant Code

Redundant code is an issue that might arise if a firm does not strategically use open source. Across different departments, functionally equivalent tools may be derived from distinct packages or code libraries. There are several packages offering the ability to run a linear regression, for example. However, there may be nuanced differences in the initial setup or syntax of the function that can propagate problems down the line.

[7] https://github.com/pandas-dev/pandas/issues/

Insufficient Documentation

Open source documentation is frequently lacking. In financial services, this can be problematic when seeking to demonstrate a clear audit trail for regulators. Users must take care to track the changes and evolution of open source programs. The core calculations of commonly used functions or those specific to regular tasks can change. Maintaining a working understanding of these functions in the face of continual modification is crucial to ensure consistent output. Tracking that the right function is being sourced from a specific package or repository of authored functions, as opposed to another function, which may have an identical name, sets up blocks on unfettered usage of these functions within code. Proprietary software, on the other hand, provides a static set of tools, which allows analysts to more easily determine how legacy code has worked over time.

Legal Implications of Open Source Licensing

Finally, it is important to be aware of open source licensing constraints and to maintain active licensing governance activities to avoid legal issues in the future.

Similar to the copyright concept, some open source creators have adopted the concept of 'copyleft' to ensure that "anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it."[8] This means that, legally, for any software that contains a copylefted open source component, whether it comprises 99% or 0.1% of the application code, the entire source code must be distributed with the software or be made available upon request. This is not an issue when the software is distributed internally among corporate users, but it can become more problematic when the company intends to sell or otherwise provide the software without revealing the internally developed codebase.

Not all open source software is copylefted – in fact, many popular licenses are highly permissive with very few restrictions. Figure 2 shows a summary of the four most popular open source licenses.[9]

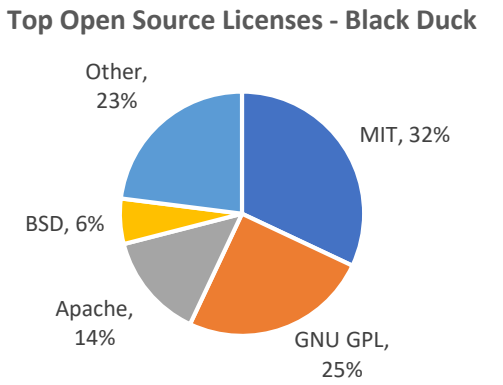
[8] https://www.gnu.org/licenses/copyleft.en.html
[9] https://choosealicense.com/appendix/

Figure 2: Popular Open Source Licenses

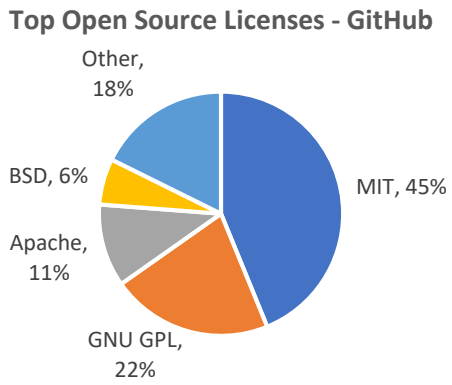
License	Commercial Use	Distribution	Modification	Must Disclose Source
MIT	✓	✓	✓	×
GNU GPL	✓	✓	✓	✓
Apache	✓	✓	✓	×
BSD	✓	✓	✓	×

Of the four, only the GNU General Public License (GPL, all versions) requires the creators to disclose the source code. Between 20% and 25% of all open source software is covered by the GNU GPL.

Figure 3: Top Open Source Licenses



Source: Black Duck KnowledgeBase



Source: GitHub

OSSRA found that 75% of applications contained at least some components under the GPL family of licenses, and that only 45% of those applications complied with the GPL copyleft obligations. Overall, the Financial Services and FinTech industries maintained 89% of all applications with at least one licensing conflict.

Most open source software, even that which is licensed under the GNU GPL, can be used commercially. For example, a company can use and internally distribute a financial model written in R, an open source programming language licensed under the GNU GPL 2.0. However, important legal consequences must be considered if the developed code will be later distributed outside of the

company as a proprietary application. If the organization were to sell the R-based model, the entire source code would have to be made available to the paying user, who would also be free to distribute the code, for free or at a price. Alternatively, a model implemented in Python, which is licensed under a Berkeley Software Distribution-like agreement, could be distributed without exposing the source code.

Open Source Software for Mortgage Data Analysis

Despite the risks outlined above, many financial institutions are embracing open source software and its benefits. We've observed that smaller entities that don't have the budgets to buy expensive proprietary software have been turning to open source as a viable substitute. Smaller companies are either building software in house or turning to companies like RiskSpan to achieve a cost-effective solution. On the other hand, bigger companies with the resources to spare are also dabbling in open source. These companies have the technical expertise in house and give their skilled workers the freedom to experiment with open source software.

Within our own work, we see tremendous potential for open source software for mortgage data analysis. Open source data modeling tools like Python, R, and Julia are useful for analyzing mortgage loan and securitization data and identifying historical trends. We've used R to build models for our clients and we're not the only ones: several of our clients are now building their DFAST challenger models using R.

A RiskSpan client recently sought to generate timely, enhanced, and transparent disclosures for pass-through securities backed by mortgage pools. We applied Apache Hadoop, an open source framework for distributed storage and processing commonly used with big data, to process loan-level and security relationship data and to perform the required daily calculations for newly issued securities. On a real-time basis, new securities were processed on the Hadoop cluster to create aggregate values and stratifications based on various security characteristics. The underlying loan-level data for these calculations was stored and retrieved from a normalized data store in a Hadoop Distributed File System (HDFS). Hadoop made processing time six times faster by enable us to generate consolidated disclosures by distributing computationally expensive calculations and millions of loan records, resulting in a 600% decrease in processing time.

While our developers worked within the Hadoop framework, our testing team used R and Python to leverage open source applications to test the disclosure outputs. Microsoft Excel, a proprietary tool, is often used to test and verify these sorts of disclosures. However, spreadsheet software, while widely available and decipherable, is slow and bulky. We leveraged R's processing power to download and pre-process publicly available Agency data and run it through Python's many relevant packages to recreate the disclosure, compare it against the one generated using Hadoop, and create a formatted

testing log that can be easily viewed in Excel or a simple text editor. The best part: the entire process can run independently alongside production within minutes, sending a summary of its activities to the testers.

Open source has grown enough in the past few years that more and more financial institutions will make the switch. While the risks associated with open source software will continue to give some organizations pause, the benefits of open source will soon outweigh those concerns. It seems open source is a trend that is here to stay, and luckily, it is a trend ripe with opportunity.

Using Open Source Data Modeling Tools

Deciding on whether to go with open source programs directly impacts financial services firms as they compete to deliver applications to the market. Open source data modeling tools are attractive because of their natural tendency to spur innovation, ingrain adaptability, and propagate flexibility throughout a firm. But proprietary solutions are also attractive because they provide the support and hard-line uses that may neatly fit within an organization's goals. The considerations offered here should be weighed appropriately when deciding between open source and proprietary data modeling tools.

Questions to consider before switching platforms include:

- How does one quantify the management and service costs for using open source programs? Who would work on servicing it, and, once all-in expenses are considered, is it still more cost-effective than a vendor solution?
- When might it be prudent to move away from proprietary software? In a scenario where moving to a newer open source technology appears to yield significant efficiency gains, when would it make sense to end terms with a vendor?
- Does the institution have the resources to institute new controls, requirements, and development methods when introducing open source applications?
- Does the open source application or function have the necessary documentation required for regulatory and audit purposes?

Open source is certainly on the rise as more professionals enter the space with the necessary technical skills and a new perspective on the goals financial institutions want to pursue. As competitive pressures mount, financial institutions are faced with a difficult yet critical decision of whether open source is appropriate for them. Open source may not be a viable solution for everyone—the considerations discussed above may block the adoption of open source for some organizations.

Governance risks are specific to how open source tools are integrated into existing operations. These risks can stem from a lack of formal training, lack of service and support, violations of third-party intellectual property rights, or instability and incompatibility with existing operating environments. Successful users of open source code and tools devise effective means of identifying and measuring these risks. They ensure that these risks are included in process risk assessments to facilitate identification and mitigation of potential control weaknesses.

However, often the pros outweigh the cons, and there are strategic precautions that can be taken to mitigate any potential issues. Security vulnerabilities, code issues, and software licensing should not deter developers from using the plethora of useful open source tools. Open source issues and bugs are viewed and tested by thousands of capable developers, increasing the likelihood of a speedy resolution. In addition, a company's own development team has full access to the source code, making it possible to fix issues without relying on anyone else.

As with any application, effective governance and controls are essential to a successful open source application. These ensure that software is used securely and appropriately and that a comprehensive testing framework is applied to minimize inaccuracies. The world of open source is changing constantly –we all just need to keep up.

References

https://www.redhat.com/en/open-source/open-source-way

http://www.stackoverflow.blog/code-for-a-living/how-i-open-sourced-my-way-to-my-dream-job-mohamed-said

https://www.redhat.com/f/pdf/whitepapers/WHITEpapr2.pdf

http://www.forbes.com/sites/benkepes/2013/10/02/open-source-is-good-and-all-but-proprietary-is-still-winning/#7d4d544059e9

https://www.indeed.com/jobtrends/q-SAS-q-R-q-python.html

ABOUT RISKSPAN

RiskSpan enables leading financial institutions to derive actionable insights by solving their toughest data and analytical challenges. Our scalable platform streamline loan data management by fusing large volumes of disparate datasets. We further arm analysts and business leaders with powerful predictive analytics to make informed data-driven decisions. As market experts in loan data management and predictive modeling, we transform your seemingly unmanageable data into profitable business assets. Our team understands enterprise data governance, helping you achieve new heights of analysis and reporting while cutting costs, minimizing processing time, and increasing returns on your investments. We not only get data, we get your data, delivering end-to-end solutions tailored to your business.

CONTACT

Headquarters
1300 17th St North
Suite 1250
Arlington, VA 22209
(703) 956-5200
insights@riskspan.com