End-to-End Open-source Collaboration Guidance

Data Visualisation & Open Source Technology Working Group

PhUSE Working Group

This whitepaper provides guidance on the use of Open-Source Software (OSS), as well as collaboration on and creation of open-source projects used by data scientists in clinical reporting workflows

Table of contents

Guidance scope and purpose

The primary aim of this collaboration is to provide guidance within the context of how open source is relevant to PhUSE members, and link out to more information to avoid duplication on more generalisable topics. In this guidance, R packages are referenced as an example OSS project that is a focal point today in clinical reporting, but the principles extend to other libraries in python, Julia, javascript etc. The following topics are covered in this white paper:

Using open source

- Relevance of different licence types
- Watchouts on governance models and assessing risk
- Landscape of tools available for vulnerability detection, validation/qualification/risk and enforcing licence policies, with particular reference to R-specific tools

Releasing open-source

- A summary and recommendation of licence types, with particular focus on permissive vs copyleft licences and the ramifications on code built on top of your project Relevance of licences present in dependencies, direct vs transitive dependencies, and the issues around compiling with dependencies that could occur in something like a public shiny app
- Landscape of places to place open-source projects and build collaborative communities
- Pros/benefits and cons/risks for companies to open-source clinical reporting codebases
- Governance models for open-source projects with reference to their use today across clinical reporting collaborations
- Survey and summary of contract types present where intellectual property and copyright is shared between companies
- Tools available to understand the general health of projects (e.g. LFX tools), with specific reference to R extensions (e.g. metacran, riskmetric, openpharma)
- Examples of release models, particularly where projects have inter-project dependencies (e.g. tidyverse de-coupled release model vs bioconductor cohort release model)
- Tools for releasing and maintaining projects, with particular reference to tools for R packages

Definitions

CLA: Contributor licence Agreement. Has a similar purpose to a DCO (Developer Certificate of Origin).

CSR: Clinical Study Report

eCRF: electronic Case Report Form

GPL: GNU General Public Licence

MIT: Common acronym for a licence released by the Massachusetts Institute of Technology

OS: Open-Source

OSS: Open-Source Software

IP: Intellectual Property

1 Open source: the what and why

'Open Source' software is software covered by a licence that legally allows access and inspection of the software's source code. The many varieties of open-source licences determine what you can then do with the software's source code, i.e. copy, modify, contribute or redistribute. Being able to view and then do something with source code wasn't always so. The term 'open source' has been in use at least since the 1990's[1] and the principles behind the term pre-date computer software. Thus, as long as there has been source code there have been efforts to make it 'open source'.

As computing systems became widely adopted in universities and beyond, so to the value of freely accessing the source code of the software they ran became apparent. This effort was described as making software 'free' by Richard Stallman and formalised by the creation of the Free Software Foundation in 1985, including the creation of a legally enforceable licences (the GNU Public Licence) to enshrine source code as 'free', that is, having the free-dom to access. Although this effort was the genesis of today's open-source communities, many people mistakenly understood 'free' to mean gratis, which was incorrect: most open-source licences allow the software to be sold for a fee.[2] As is the case, even if the main goal of open source is not creating software gratis, it so happens that the majority of open-source software is made available at no cost. Regardless of whether it is sold for a fee or not, the term 'open source' is the preferred term by most with respect to software with a licence that allows access to the source code.

Readers coming from the pharmaceutical industry probably perceive a contradiction here: how can software which is typically gratis to use, have any intrinsic value to either business or private users? Fair enough: this industry depends on capital investment which then depends on retaining the details of their drugs and production secret. The difference lies in the utility of (some) software, versus, in this example, a drug or therapy. Certain categories of software enable the creation of new value. Obvious examples being programming languages enabling creation of specialized applications which can support a specific business process, e.g. C, Python, R and many others. The ability to use and improve these open source languages freely accelerate in multiple dimensions the ability to create business value, e.g. specialized smart phone apps that offers a service to end-users. Imagine if you have an idea for a smart phone app, but before you can write a line of code, you need to buy a licence to install that language. And after investing the time and money to access this language you realise it doesn't work as well as you need for your particular app. Or worse yet, it has a bug which renders it unfit for your purpose. Little chance you can resolve this quickly. Open-source software does not have

these restrictions so you can focus all your resources on end-user value, not the tools needed for creation.

Open source is also a step towards insuring repoducibility. Consider an analysis done in a propriety language by a pharmaceutical. An academic accessing the same data through a data sharing initiative may find results that contradict those done using the propriety tool. They would not be able repoduce the results - and in fact any attempt to repoduce results is dependent on the for-profit company providing a license to generate that insight.[3]

The drugs and therapies manufactured by the pharmaceutical industry are the equivalent of a smart phone app: they provide end-user value. It's sound business logic to open source the tools used to create these products: remove the restrictions to creating drugs and enable each company to sharpen their focus on developing and delivering them.

2 Using open source

2.1 What is the open-source health of the package?

The communities that maintain and build open-source packages are diverse, and there are no set conventions on how they are maintained, resourced, and governed. There are no universal magic metrics to summarise whether an OS project is *'healthy'*; for example if a project has had no activity for 12 months, is that because the product has been abandoned/superseded, or could it be it had a small well-defined scope and is now stable and feature complete? The following section is a non-exhaustive discussion of topics relevant when using open-source data science projects.

2.2 How can I see the activity of an open-source project?

Many, but not all, open-source projects are on github.com or gitlab.com. On github.com, every repo contains a tab called Insights, from where you can see information on the people who contributed lines of code to a project. Of a particular interest might be the Contributor tab within Insights, an example screenshot of the dplyr R package contributor page is in Figure 2.1.

Some sites like openpharma.pharmaverse.org (specific to R and python packages in pharma) and OSS Insights (https://ossinsight.io/; powerful tool for any project on GitHub) also provide more specific insights into the community engagement behind each project hosted on github.com.

2.2.1 How active are the community behind a project?

The activity on a project does not tell you the quality and extent of use of a project. Two examples are:

- A project could have almost no active community in terms of recent contributions or response to issues, much like the R package survival (https://github.com/therneau/survival), yet be a stable and critical package in R installations.
- A project could also have no activity as it has been abondened after or before it reached v1.0.

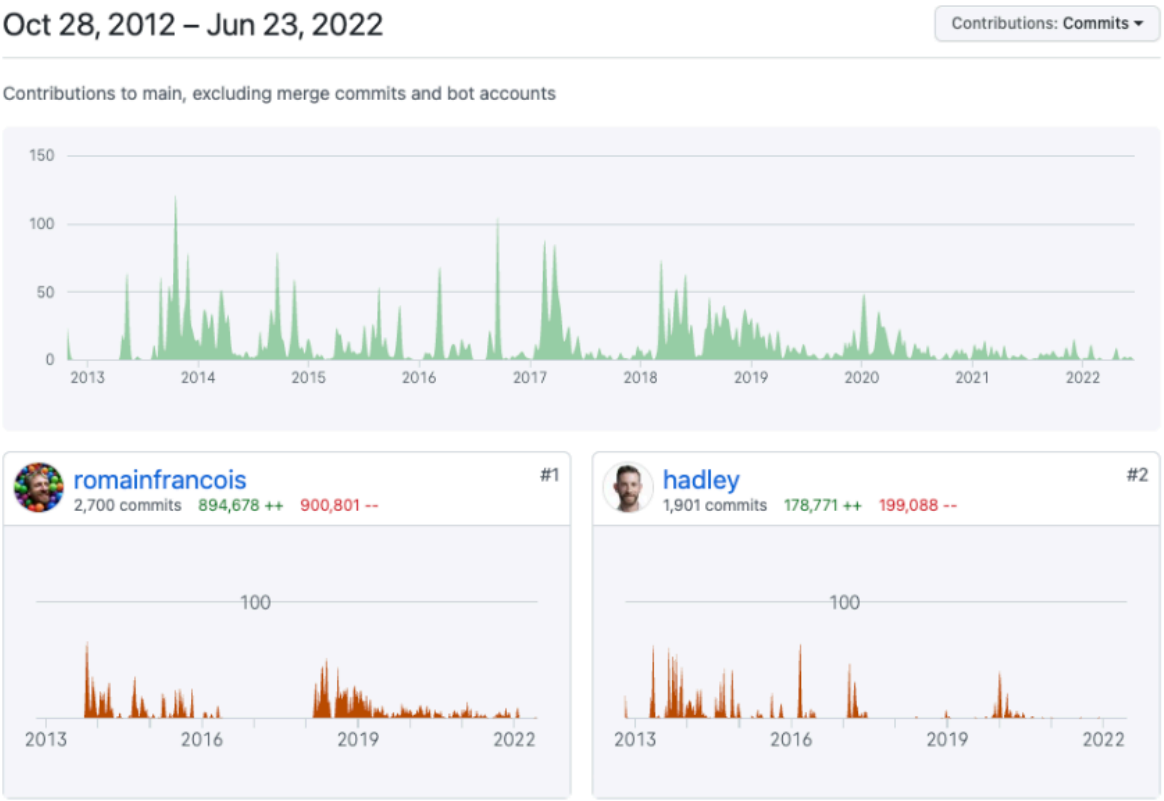

Figure 2.1: Screenshot from Insight tab for the dplyr R package

The community behind a project is also not limited to the people that contribute code. Users can also engage with a project via giving feedback via mechanisms like GitHub issues, emailing authors or engaging in discussions on GitHub issues. Figure 2.2 is an example of an issue page for the teal R package. The figure shows that teal has 24 open issues, and 266 closed issues. Small speech bubbles on the right of the figure show discussion have occurred on some issues. By looking through the issues, subjective impressions on community health can be made, for instance whether it's a few people giving feedback and one person developing, does it have stale issues no-one replies to, or does it have a lively community engaged in discussion and coordination.

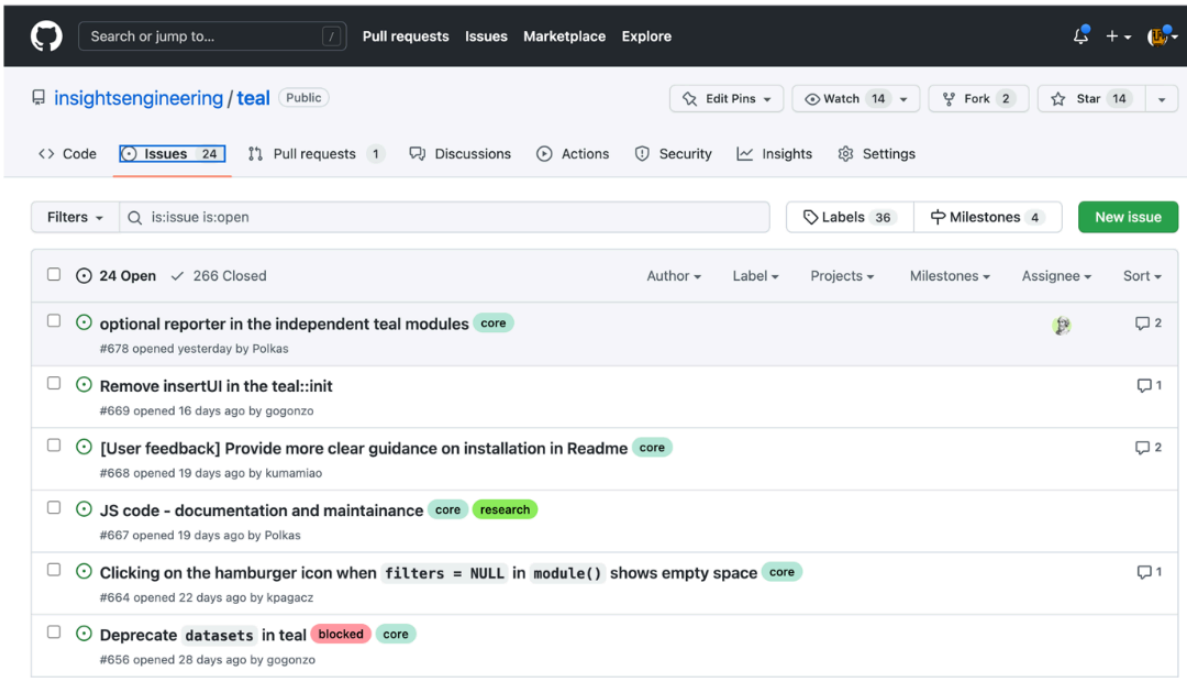

Figure 2.2: An example screenshot of the R package teal's issue page

Packages can also be open sourced without having the place they develop the code exposed to the general public. An example is the randomForest package (stat.berkeley.edu/~breiman/RandomForests), which is an open sourced (GPL-2/3) R package where the source code of the releases is open sourced for use, but the package authors do not give users access to view the place where they develop code. This does not mean the quality of the code is inferior, but does indicate there is an additional barrier to engaging with the package development as the first step would be to contact the authors.

Some things to consider when trying to establish the activity of a community are:

- How many individuals contributed to the project?

- What is the spread in contributions? What is the size of the 'core' group that contribute the majority of the code? What is the spread of commits – is it highly skewed to 1 or 2 people contributing?
- What is the recent and trends in commit activity? Is it currently active, formerly or is yet to become active?
- How many open and closed issues are there? If it's a low number, is that in line with the age and expected use of the project?
- Are there 'stale' open issues, where issues remain open for months or years? Are many of these stale issues with comments, suggesting some discussion, or absent of comments suggesting there is no feedback loop present between issues and the codebase? A thing to also look for is whether closed issues are resolved, as some projects use bots to automatically close stale issues.

2.3 How do I find open-source projects?

Numerous methods exist to find projects. Specific to R projects, the following sources exist:

- pharmaverse.org: opinionated/curated effort to provide end-to-end tools for clinical reporting.
- http://openpharma.pharmaverse.org: un-opinionated tracker of packages built by pharma for pharma use cases. It also and indexes and provides package metadata in a dashboard, and provides metadata to pharmaverse.org.
- The R universe hosts ecosystems of packages in CRAN-like repositories. As an example, the Pharmaverse has the 'bleeding edge' of the main branches of all included R package available as a CRAN-like repository.
- rseek.org: Google filter for R relevant content.
- rinpharma.com/publication: the proceedings of the R/Pharma conference contain many relevant projects.

2.4 What do I do if I see a project I could use, but it needs modification for my use case?

Using R packages as an example, if your analysis plan requires creating a Kaplan Meier plot, you could implement this using open code you program using R base plotting functions. Alternatively, you could introduce a dependency on a package that provides that functionality as a parameterised function, like survminer, visR or tern. Occasionally an existing package

may be missing a feature you want, as can be derived from the presence of at least 3 R packages with a Kaplan Meier plotting function. In such cases, you may need to extend, or start a new package.

When an existing tool is not a perfect fit, it can be difficult to decide whether to extend an existing package, or whether it may be worth starting a new one. Some resources to help understand how to contribute to a new package are:

- A blog post by Jim Hester on contributing to the tidyverse
- Many packages have a CONTRIBUTING.md file, or mention in the README.md, how you can contribute. They may also be a dedicated tag for issues discussing new features (e.g. `'enhancements'`).

2.5 What can help me understand the risks around using an open-source project?

Risk can come from several domains including;

- Security, e.g. it has malicious code,
- Quality, the package has poor documentation and code is unreliable.
- Accuracy, the package does not correctly reference what it does, or implements it incorrectly.

The R validation hub is a pan-pharma organisation, that aims to coordinate between pharma companies how the validation (and by extension risk) in R packages is undertaken and documented. Of particular relevance is the Case Studies repository, which contains examples from Roche, Merck and Novartis (as of July 2022) on how they approach validation and risk mitigation. The R Validation Hub also created `riskmetric` as a tool to extract metrics relevant to validation, and is continuing work on the Risk Assessment App, which aims to provide an application that will surface these metrics to a user to help evaluate an R package.

Roche has also open sourced a github-action called thevalidatoR, which is available on Github Marketplace, which will generate a PDF with the unit testing results, as well as a traceability matrix of documentation against tested functionality in a specified container.

The `oysteR` R package can help scan R projects for known vulnerabilities via a REST API interface into the vendor tool *OSS Index* from *sonatype*.

2.6 Licences: using a project

The licence of projects you depend on, particularly if you incorporate the source code into your compiled/shared product, can have drastic effects on what you can do with your project. It is always important to seek in-house counsel advice on your companies position on different licence types.

As a general guidance:

- There are permissive licences that allow people to use a project in almost any way, through to copy-left licences that prevent distributing and, in some cases, monetizing any project that incorporates the dependency into its codebase.
- Two key resources to understand licence types are https://choosealicense.com/ and https://opensource.org/licenses.

3 Releasing open source

Without open-source, many of the R packages we use today would never have developed or would be kept behind company firewalls. Open-source provides a mechanism for code sharing and collaboration, which in turn means talent can flow from company to company across our industry, we prevent duplication of the same post-competitive tools, and we move closer to decrease the burden on reviewers by bringing consistency in both our code and outputs in a submission.

3.1 Intellectual Property (IP)

IP is often bucketed into pre-competitive and competitive IP[4], with post-competitive being a less established term we will define in this guidance. In clinical reporting, we place significant resources into the collection and presentation of information that was collected on our competitive IP in confirmatory clinical trials. In order to help separate this simpler case from pre-competitive – here we define as post-competitive a unique scenario of code that takes data generated as part of confirmatory studies (e.g. a Phase III trial) and creates an output. Post-competitive IP is where the benefits of open sourcing and encouraging between company collaboration can be more clearly differentiated from potential competitive advantage in developing new medicines.

The following summarizes the three types of IP:

Pre-competitive IP which is not a competitive advantage. This can be a complex definition, and will require guidance from company council. For instance, data standards may clearly be pre-competitive, but for anonymised data from historical trials, or an algorithm that generates risk scores for a certain outcome could provide a competitive advantage, or be defined as pre-competitive.

Competitive IP Clinical reporting relevant examples would be information on a new target, molecule or algorithm that provides an advantage in the creation of new medicines, or as a standalone data product that can be monetized.

Post-competitive IP A less common term we have defined to be where code collaboration improves the efficency of insights, rather than the creation of insights that would otherwise not be possible. In the context of PhUSE collaborators, this includes packages that take

CDISC data and apply templated data steps and visualizations to prepare a CSR, like those seen in the pharmaverse.

3.2 Preparing for release

3.2.1 When is a good time to open source?

As a general rule *arising IP*[5], that is IP generated as part of the project, is simpler to handle than *background IP* that already exists. There is often a benefit to define what you want to do, decide if it would be open sourced, and if so, start it in an open-source setting. This also helps to encourage defining a clear scope from day one, and encourage others to engage early rather than initiate additional projects that later may not be compatible without significant re-factoring.

3.2.2 Does it matter where you put a package on github.com?

What are the differences between GitHub organizations that host packages like phuse-org, rinpharma, ropensci, openpharma, pharmaverse, pharmar, personal organisations, company owned organisations and organisations created to host a single project?

Ultimately, the licence chosen has an impact on how a package can be used, rather than the location the code is shared from. The location though can influence how a project is perceived. If it is hosted on a GitHub organisation with the name of a pharma company, relative to a pan-company organisation, it may imply that the project is '*Company A's*' project rather than something they wish to co-create. As a general rule, the recommendation would be to place it in a company's organisation if you wish to remain control of the roadmap, but look to pan-company organisations if you wish to co-create and co-own the packages trajectory. Some examples are;

- Personal Github orgs
  - diffdf (gowerc/diffdf) and survival (therneau/survival) are examples of two repositories used in pharma hosted in Github orgs belonging to a specific individual.
- Project/Initiative Github orgs
  - openpharma: While openpharma has a dashboard and metadata pipeline that is agnostic to where a package comes from, it also will house packages that do not want to be associated with a specific company or organisation.
  - pharmaverse: A sub-set of the pharmaverse clinical reporting repositories are also hosted on the pharmaverse Github org.
  - pharmaR: Houses repositories from the R Validation Hub working group.

- Company Github orgs
  - Many companies maintain Github orgs either at the company or department in a company level, like GSK-Biostatistics, Roche, Genentech, Novartis
- Organisation Github orgs.
  - phuse-org: PHUSE projects and working groups from PHUSE.
  - ropensci, ropenscilabs, ropensci-docs, etc: rOpenSci maintains several GitHub orgs, with rOpenSci housing mature R packages contributed by their staff, or peer-reviewed.

3.2.3 What is important to look for when releasing a package that started life internally?

If a package started its development on an internal git server, or a private repository on github.com, there could be some risk of exposing data either in issues, or historical commits. These could range from screenshots of patient data, tables or other business confidential information in issues, to passwords or files in the git commit history that were deleted but not purged. The recommendation is to always flatten the commit history, and wipe issues by starting a new git repository when open sourcing unless you are certain no information can be leaked.

3.3 Could others claim we stole their IP?

When discussing the open sourcing of a codebase, it is important to flag to internal counsel existing external projects, and the overlap of scope with the project you intend to release.

It is possible that decisions made before open sourcing could become a risk after open sourcing. As an example of a plausible scenario; a team need to implement a new function. This function exists in another GPL-3 copy left licenced project. To add that project would introduce multiple dependencies that aren't used by that particular function so a member of the team decides to copy the function into the package. One year later, the package is open sourced with the licence infringing code. Such an occurrence could be lessened by a Contributor Licence Agreement (CLA; see the bot contributor-assistant for an example of CLA automation). A CLA helps ensure that anyone contributing to a project acknowledges specific terms expected of contributions, like the contributions are novel code and the author will abide by the projects licence terms. In the absence of a CLA it is important to ensure that all code within the package is original, and there is no culture of cannibalising external code and infringing on people's copyright within the development team even for internal projects.

3.4 Reputational risks and supporting others

What are the expectations when I release a package? Are there risks to my company's brand having abandoned non-maintained packages?

In this guidance it is suggested to open-source early, yet doing so could expose projects that are not ready for use, might be cancelled before reaching v1.0 or are never successfully adopted. The ratio of failed to successful projects is an important consideration, but a skew in that ratio being a negative indicator can be mitigated if repositories are clear on what stage of the product life cycle they are at and make use of tools to inform users if a project has been deprecated, or are looking for new maintainers to take over the project.

3.5 Licences: releasing a project

Ultimately, the licence used for a project would require in-house counsel guidance on what licence is preferred.

All code open-sourced should have a licence. The licence has a standard location of being a text file called 'LICENSE' in the root of the project folder, a text file called 'LICENSE.txt', or a markdown file called 'LICENSE.md'. Of particular note is that R packages often have the licence specified in the R specific location of the DESCRIPTION file, or may have it in both the standard and R specific locations (in rare cases these can also contradict so it is important to read both).

Generally, permissive licences are more common in clinical reporting, with the majority of pharmaverse R packages using an MIT or Apache 2.0 license. These licences allow distribution, commercial use and modification. One primary difference between MIT and Apache 2.0 is that the latter has patent protection language and rules around trademark usage, and may be preferred in larger projects due to its focus on more explicitly spelling out the terms.

As a general guidance, if the purpose of the project is to let future contributors freely use the code, MIT license is a concise permissive license to adopt. In the pharmaceutical industry, however, the patent of the code is often of concern in a post-competitive environment across companies, and thus an Apache 2.0 license could be more suitable. On the other hand, the copyleft license (e.g. GPLv2, GPLv3) demands any downstream derivatives to follow the same copyleft license of the source project and generally should be avoided. Sometimes, a company's legal team might come up with their own license that is not listed as one of the approved open-source licenses. It is highly recommended to only use standard open-source licenses, as these are verified by the Open-Source Initiative, so others can easily understand the governance model of your project.

A licence is ideally one of the first commits made at project initiation, because a change in the license could impact many aspects of the project. With a permissive license, others have

been granted permission to license modification from its inception. When under a permissive license, you could change to a license with more requirements, but this would not rescind the historical codebase that has a more permissive license.

3.6 Collaboration and governance models

Open-sourcing a project allows others to leverage the code, but the ultimate goal is often that the open-source community adopts and helps extend and evolve the project. How projects govern this shared development is diverse. A commonality across all projects is that the repository, and it's `main`/production branch, will have some form of write access control, meaning a level of governance is present even if it's not formalised.

3.7 What different governance models exist for open source?

There is no definitive definition of open-source governance models. The following models are based on mapping Redhat, opensource.com and Linux Foundation notes to the packages relevant to clinical reporting.

Single Entity This category refers to a project where a single entity is the final decision maker, regardless of whether that single entity is an individual, a company or other legal entity. This governance model is sometimes referred to as the "privately open source", "founder-leader", or "benevolent dictator" model. The single entity controls which pull requests go to master and provides instruction on how new code should integrate in order to be accepted. Famous examples are Python until 2018 and Linux. Within pharmaverse.org, diffdf and many of the single company governed packages are an example of this governance model.

Steering Group This category refers to a project where the ultimate decision-making capacity is shared between more than one entity. The structure of the group and manner in which the group makes decisions can vary. The name used to refer to the group can also vary, examples include "governing board", "steering group", and "council". A famous example includes the relatively oligarchical Python Steering Council from 2018, however many projects prefer simple democracies, or merely that a specific number of approvals from among the contributing entities are required to approve acceptance to the production branch. Within pharmaverse.org, admiral is an example of this governance model.

Do-ocracy This category refers to a project where access to the production branch is given out fairly freely, usually based on prior interactions with the primary contributors, or actual contributions via external pull requests. Trust is placed in the community to come to an agreement regarding acceptance to the production branch. This category

is sometimes also referred to as a "self-governed" or "non-governed" governance model. Within pharmaverse.org, visR is an example of this governance model.

Foundation governed A legal body (e.g., non-profit) assumes control - an example organisation is the Linux Foundation which governs many projects, while in Pharma there are parallels to efforts like Transcelerate and OHDSI. There are no examples of this model within pharmaverse.org, but R/Pharma repositories do follow this model, where the registered non-profit Open Source in Pharma governs the github organisation.

If two or more companies want to formally collaborate on an open-source project, what is the role of legal contracts between the companies when the code is open-source?

Contributions to open-source code can come in many forms, and there is a great deal of diversity in projects relevant to clinical reporting. This is an emerging area for pharma companies, and so we will focus on promoting awareness, rather than giving firm guidelines.

3.8 When do we need contracts?

When initiating a project like an R package, or when another company is considering investing in collaboration to an existing project, there could be a discussion on having a legal framework layered on top of the collaboration. To help contextualise this, we will use four example projects.

dplyr The dplyr project is a ubiquitous in pharma, but is a generic data science package for data munging. The code owners are listed as an individual from a vendor, academia and a consultancy and it's released under a permissive license. This package is extensively consumed, and a core dependency in data related packages like admiral. This package is heavily depended on Pharma, but no legal agreement exists beyond the permissive licencing on the project.

gt There is a large spread of table generation packages in Pharma, but several Pharma companies (Roche and GSK – add links to their issues before pub) have publicly been exploring extensions that would allow the use of gt in TLG generation for CSRs. No legal agreement exists beyond the permissive licencing on the project.

pkglite Submitting code to the FDA requires collapsing the contents into text files with restrictive formats. pkglite exists to collapse and reconstitute an R package before and after the eCTD submission portal. pkglite uses a copy-left license, and copyright is owned by Merck. No legal agreement exists beyond the copy-left licencing on the project.

admiral admiral is an R package for creating ADaM datasets. The copyright is held between Roche and GSK, and it is permissively licensed. A contract exists between Roche and GSK on their collaboration model. Other Pharma's have contributed and offered to extend admiral without legal contracts in place on the original codebase.

The examples above were intended to highlight that the majority of R packages used by Pharma companies are done so without legal contracts in place, beyond the license of the project, even when some collaboration takes place.

It remains a discussion point though whether licenses are required, and the decision to create a license may become relevant if companies want to formally pool resources. It's important to note that with permissively license projects, it is possible that if two entities want to take a package in different directions, they are able to by forking the project. So, contributions to another entities package are not lost to the contributing company.

3.9 The risks of open-sourcing

One open question is often how does open-sourcing open a company up to liability, indemnity and warranties. We previously discussed CLA bots, as a mechanism to reinforce the need for contributions to be original, and never cannibalised from another project. For remaining risks from others using an open sourced codebase, licenses will include some language. As an example, 50% of the MIT license is devoted to this topic with the following working:

> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4 Contributors

We'd like to thank the following people that have contributed initial content or revisions, in alphabetical order:

Anders Bilgrau, Novo nordisk

Estella Dong, Bayer

James Black, Roche (co-lead)

Karl Brand, Bayer

Keaven Anderson, Merck

Michael Stackhouse, Atorus Research (co-lead)

Ross Farrugia, Roche

Samir Parmar, Pfizer

References

1. Christine Peterson. How i coined the term open source. Published online 2018. Accessed February 2, 2023. https://opensource.com/article/18/2/coining-term-open-source-software
2. OSI. History of the OSI. Published online 2018. Accessed February 2, 2023. https://opensource.org/history
3. Bruno Rodrigues. Open source is a hard requirement for reproducibility. Published online 2022. Accessed February 20, 2023. https://www.brodrigues.co/blog/2022-11-16-open_source_repro/
4. Contreras J. Pre-competition. *North Corolina Law Review*. 2016;95(1):Article 3.
5. Law Insider. Arising intellectual property definition. Published online 2022. Accessed February 2, 2023. https://www.lawinsider.com/dictionary/arising-intellectual-property/