

Opportunities and Challenges for R as an open-sourced solution for statistical analysis and reporting, from vendor's perspective

Peng Zhang, Lizhong Liu, and Tai Xie, CIMS Global, LLC, Somerset, NJ, USA

ABSTRACT

In recent years, there is an increasing trend from using SAS® to using open-sourced R for statistical analysis and reporting. Several large pharma companies have conducted mock submissions to FDA. This improves the progress of further open-source applications in clinical trial reporting. While most of experience that we can borrow are from large pharmaceutical companies, there are rare literatures and materials about how to start R practice from small biotechnology companies, CRO, or data service companies, e.g. what can be done through R, team building, training procedure, R development focus, system environment and application of R in analysis for clinical trials. In this paper, we will review current practices of R and share our approach about standardization, automation, and further consideration in pharmaceutical industries.

INTRODUCTION

When it comes to using open-sourced R, users can easily start with a local R and RStudio, accessing the package and potential data, following by solution by using corresponding packages. The results can be attractive by using {ggplot2}, {rtables}, {teal}, etc. While those can be achieved for exploratory purpose, additional need comes out for regulatory compliance, GxP, CFR Part 11, etc. While there are lots of literature about how to make it standardized, many paper provided from Sponsor perspective, though some of papers from vendor perspective but only a few. We would like to point out the difficulties we faced while starting transitioning to R solutions and what practices we faced at this new era.

At the meantime, it comes to a challenge for a company to start to build R team, while previous practice focus on SAS approach. The mature development of SAS macro has been established in recent years. While we consult lots of R team leaders from big companies and pharma, they can usually have different answers due to the various situations. We provide our solutions below as a reference for a potential approach on how to prepare well on statistical service in R.

GET FAMILIAR WITH R IN PHARMACEUTICAL INDUSTRY

It is necessary for a team leader to know the current development of R. While the shift from previous SAS practice to current R has becomes apparent, the usage of R has been transitioned from traditional way (e.g., R base, plot()) to modern usage (e.g., {tidyverse}, {ggplot2}), and even more trial-related packages (e.g., {admiral}, {rtables}, {teal}, {tern}). While all the approaches can achieve the goal, the efficient arrangement of package and function learning decides the progress of the team development. Below we provide a necessary start as a reference.

Forum and Organization

The reference below serves as the most active forum we refer to our daily practice for their recent constant development and update:

- Pharmaverse: <https://pharmaverse.org/>
- PharmaSUG Papers in recent years: <https://www.pharmasug.org/us/2024/papers.html>
- PHUSE Papers and presentation slides in recent years: https://phuse.global/Communications/PHUSE_Archive/1
- R consortium and webinar: <https://www.r-consortium.org/webinars>
- R Validation Hub: <https://www.pharmar.org/>
- Openstatsware: <https://www.openstatsware.org/>

Online R learning

Atorus Academy (<https://academy.atorusresearch.com/dashboard/>) provides around 2-week study R course about transition from SAS to R in the pharmaceutical usage, with a membership-based approach.

Results between SAS and R

CAMIS (<https://psiaims.github.io/CAMIS/>) records the statistical results difference between SAS and R and focus on not only the consistent methodology but also digits number for each function into details. The website has come out in recent years and can be a good reference.

CONSIDERATIONS IN R USAGE

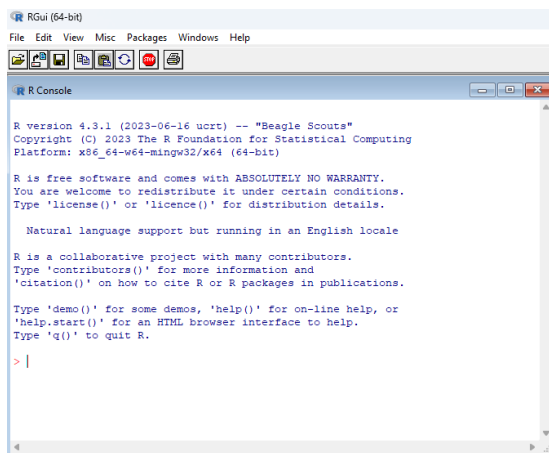
In the pharmaceutical industry, it is important to derive consistent statistical results with replication. Based on current practice in SAS, one would expect the same results under the same SAS version, even if they may be installed under different servers, operating system, etc. However, when it comes to R and open-source, there are several factors to be considered, from system level (IDE, R version, R package version) to study level (project-level R package, program validation, regulatory submission). By communicating with statistical programmers and statisticians from different areas including North America, European area, Asia, etc., people can have different background of R usage and practice. We summarize our understanding below for reader's reference as a potential benefit to organize one's internal preparation.

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

There are three IDEs that are usually used in current practice.

R

This should be the first IDE when user download R as it comes instantly after R is installed. This IDE (Display 1) comes with a concise interface, providing console and editor (where user can save the code). User can usually conduct statistical analysis under this IDE using R base and recommended package (See section 2 for package list in <https://www.r-project.org/doc/R-FDA.pdf>). However, this IDE is not inclusive enough for user-friendly access to new functionality (e.g., project setup, Git, Github setup, RShiny feature, etc.)

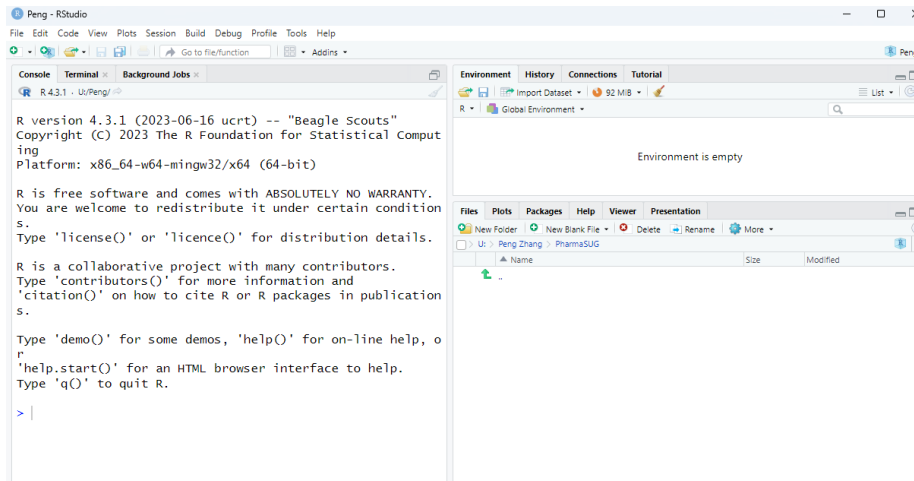


Display 1 R IDE

RStudio Desktop

RStudio (<https://posit.co/download/rstudio-desktop/>), is the most common IDE (Display 2) that modern R user currently used for their daily work and study. As mentioned, user can have a relatively better experience on finding corresponding functionality, including easy access to the files in the folder, window for visualization, package list view, internal help page, etc. Moreover, some modern functionalities are

included for project-level management, package build tools, Git and Github functionality, R Shiny application display, etc.



Display 2 RStudio IDE

Posit Workbench

We would like to mention that Posit is a new name for RStudio from year 2022 (<https://posit.co/blog/rstudio-is-becoming-posit/>), so one might expect the same or similar things when people talk about RStudio Professional Product and Posit Product.

For Posit Workbench, it provides similar, but more features compared to RStudio. Below are the key and common features from Posit:

- Browser accessed instead of local installation, preventing exposing data to desktop.
- User access controlled by SSO, limit resources by user or user group.
- Single platform with multiple R (or Python) versions to be used, and with corresponding IDE (Rstudio, Jupyter).
- Multiple sessions to be used at the same time, improving multi-tasking.
- Feature limits by admin by turning on/off 'file upload/download', self-package installation.
- ODBC Driver, allowing to pull in data from database.
- Admin UI/dashboard to manager user and sessions.

More detailed information can be found (<https://docs.posit.co/ide/server-pro/user/rstudio-pro/getting-started/>). This can be a recommended and convenient platform for organization's internal management and version control, which we will discuss in the later section.

R VERSION

With the development of R packages and more needed for daily practice, R is developing and updating R versions due to new packages coming out or new features published (e.g. RShiny, Quarto). The new version aims to keep the same functionality as previous R version, and to be compliant and supportive to the new packages. Organizations often need to do regression testing to make sure the previous features successfully derived from previous version can experience the same in the newest version. It is suggested to install multiple R versions (https://www.lexjansen.com/phuse-us/2021/as/PRE_AS01.pdf) rather than updating R version to replace previous version for the work, to at least have backup and solution for extreme cases. This procedure should be better recorded into the organization's SOP for efficient management and maintenance.

R PACKAGE VERSION

One of the most attractive features in R is that the developer can contribute their own packages to CRAN (<https://cran.r-project.org/>) beside the R base and recommended packages mentioned above. For a

common practice, user find necessary package they might need and install by using `install.packages()` or `devtools::install_github()`, without paying too much attention on the package version. Indeed, for all the packages, developers can have constant development and can have updating features into the package, which a new version of the package can be published to include those updating features. Note that for new version of package, it can include 1) new features/functionalities coming out, 2) update for existed functionality with different input, output, or procedure, or 3) depreciation of existed functions. Therefore, by using a new version package, one might have potential to experience different outputs from previous code or even issues/errors. While this can be a concern for an organization, there are some recommended maintenance procedures for package installed, to decide R package list, picking snapshot date, installation, qualification document, etc.

<https://solutions.posit.co/envs-pkgs/environments/validated/>

<https://www.pharmasug.org/proceedings/2022/SI/PharmaSUG-2022-SI-057.pdf>

<https://github.com/philbowsher/Creating-an-Internal-Repository-of-Validated-R-Packages/blob/main/Creating%20an%20Internal%20Repository%20of%20Validated%20R%20Packages.pdf>

https://github.com/philbowsher/phuse-2019-r-packages/blob/master/How%20Do%20I%20Pick%20a%20R%20Package%20for%20My%20Clinical%20Workflow_.pdf

A risk assessment approach is recommended to include external packages.

<https://www.lexjansen.com/phuse-us/2019/tt/TT11.pdf>

<https://www.pharmar.org/risk/>

PROJECT-LEVEL R PACKAGE

There is one attractive feature in R is to build R project, to make connections between files, codes, folders within such object. Some additional features can be added to R project compared to R functions including Git and Github for version control, R package development for further usage. If one would like to take advantage of this format, it is crucial to describe what package and package versions they are using for further development, as usually package development involves unit testing and validation. To make sure the results can be compared, the consistent functions and versions should be guaranteed for this procedure. It is common that developers spend multiple days figuring out why the same codes cannot derive same results from two machines due to functions used from different packages, or packages from different versions. To deal with such case, {renv} package provides an efficient workflow to have consistent package lists used and versions (<https://rstudio.github.io/renv/articles/renv.html>).

PROGRAM VALIDATION

Based on practice using SAS to derive dataset and statistical results, it is common that two programmers serving as primary and secondary role, to write code independently according to specification, derive the same results (including digits, format) and have them compared before moving into the production. In R, while this can be also achieved using similar format, an alternative solution can be using {testthat} to achieve validation under project level. While this can be used more when developing R package, {testthat} provides an efficient solution for validation of primary code.

REGULATORY SUBMISSION

While we are under potential transition from SAS to open-sourced language, there are multiple pilot studies submitted to FDA (<https://rconsortium.github.io/submissions-wg/>):

Common TFL analysis

<https://github.com/RConsortium/submissions-pilot1>

<https://github.com/RConsortium/submissions-pilot1-to-fda>

Shiny Application Submission

<https://github.com/RConsortium/submissions-pilot2>

<https://github.com/RConsortium/submissions-pilot2-to-fda>

ADaM Dataset Derivation

<https://github.com/RConsortium/submissions-pilot3-adam>

<https://github.com/RConsortium/submissions-pilot3-adam-to-fda>

Take pilot study 1 for TFL as example, the package of code and outputs come with ADRG (Analysis Data Reviewer's Guide, e.g. <https://github.com/RConsortium/submissions-pilot1-to-fda/blob/main/m5/datasets/rconsortiumpilot1/analysis/adam/datasets/adrg.pdf>), specifying the reference, source data, study information, etc. While the pilot studies utilize the format of R package for submission, the ADRG includes the operating system, R version, package information, referenced package information, submitted package folder structure, and instruction for execution or R program. The instructions will include how to install R with corresponding R version, setting up environment including directory, R package repository installation. Following this, one should carefully note the necessary information with respect to operation in R, so that FDA reviewer can replicate the results accordingly.

COMPLIANCE CONSIDERATION

There are multiple documents available from Posit as statements for the compliance discussion.

Discussion of R regulatory compliance:

<https://www.r-project.org/doc/R-FDA.pdf>

Discussion of Posit software regulatory compliance:

https://www.rstudio.com/wp-content/uploads/2019/06/rstudio_compliance_validation.pdf

Discussion of Posit-developed package compliance:

<https://www.rstudio.com/assets/img/validation-tidy.pdf>

<https://www.rstudio.com/assets/img/validation-shiny-rmd.pdf>

More information can be found <https://solutions.posit.co/envs-pkgs/environments/validation/>

Following GxP, one should consider aspects of the 1) R and R version, 2) R programs and corresponding packages, and 3) Posit Products. As mentioned above from system level and project level, the organization might want to prepare corresponding SOP to maintain efficiently and effectively and develop corresponding documentation for the procedure conducted.

CURRENT AVAILABLE SOLUTIONS IN PHARMACEUTICAL INDUSTRY

Following Pharamaverse (<https://pharmaverse.org/>), the colleagues from pharmaceutical industry are contributing open-sourced solutions (i.e., R packages) to streamline the data to results process.

RAW DATA TO SDTM

There is one current working solution of transforming from CDASH standard to SDTM CDISC standard <https://www.cdisc.org/oak>. The completion of proof of concept is expected to be mid of 2024 (https://phuse.s3.eu-central-1.amazonaws.com/Archive/2024/Connect/US/Bethesda/PRE_OS09.pdf)

SDTM TO ADAM

There is a popular procedure to automatically derive ADaM dataset based on SDTM dataset. Note that the programming usually comes with specification, the current available solution will be described as below.

1. {metacore} package read the information from specs and save them into an 'metacore' object.
2. {haven} package can read '.sas7bdat' dataset into R.

3. {metatools} package can use the 'metacore' object to derive predecessor variables from dataset read from {haven}.
4. {admiral} and admiral series package will allow an efficient derivation for each assigned variable.
5. {metatools} use the information 'metacore' object (e.g., format, length, etc.) to apply a check procedure on the final data.
6. {xportr} export the data in R into '.xpt' format dataset. Note that each there is write_sas() function from {haven} package for '.sas7bdat' format data derivation, the function itself cannot have a reliable output (https://haven.tidyverse.org/reference/write_sas.html)

The procedure above provides a solution of SDTM to ADaM process including automation way or self-coding way. To utilize the {metacore} and {metatools} correctly, one should carefully understand the function and define their organization's specs to be effectively read into metacore object. From another way, if one does not intend to automate the process, by solely utilizing {admiral} and/or {tidyverse}, the dataset can be derived based on developer's understanding of the specs. However, the functions from {metatools} and {xportr} can provide a good assessment of the quality of the dataset derived in R and can help prevent discrepancy between working dataset and expected output.

ADAM TO TFL

For this process, lots of organizations has developed their own solutions and have them open-sourced for further user to use. Leading packages include {rtables}, {Tplyr}, {tidytlg}, {tfrmt}, {ggsurvfit}, {tern}, to help developer to derive the formatted tables efficiently. The solution will usually have a pre-defined layout and make sure such layout can be applied on the data. Each solution has their focus and user can choose their preferred method as their potential foundation.

There is also developing package {cards} with respect to potential usage of ARD (Analysis Result Dataset), which help automation between ADaM and TFL and store the information.

INTERACTIVE TOOL AND RSHINY

It should be noted that an interactive tool of RShiny comes out in recent years to help simply the process of making interactive tool. Compared to traditional formatting of developing an interactive tool, which will include the parties of product, software engineer, statisticians, quality assurance, etc., such methodology allows statistician or statistical programmers themselves to develop an internal or public application for user to examine data or derive output. Current framework can include {shiny}, {golem}, and {rhino}, depending on the complexity of the application. In pharmaceutical usage, {teal} can be the most popular package, together with connection of {rtables} and {tern} for deriving TFLs, to derive interaction tool for clinical usage on examination and exploration.

While R Shiny can be a convenient tool for statistical programmers to develop in R, scalability and production can be usually a concern, as this practice does not follow a good product procedure. Usually for an application development, it will include at least the documentation of requirements, functional specification, design specification, installation and/or operating qualification, and later formal validation. Form this way, one can guarantee the traceability and the application can be consistent to what they should be. The trend of simplifying the application process will bring other questions when it comes to the later stage. However, some applications, with small, required functionalities, can sometimes be helpful for a quick walk through.

INTERNAL DEVELOPMENT CONSIDERATION

DIFFERENT AREA OF STATISTICAL SUPPORT IN R

In addressing the challenges of TFL, a crucial concern revolves around the need to streamline processes and provide a unified solution adaptable to diverse requirements. Our examination of traditional approaches to ADaM data processing into TFL format unveiled a repetitive and time-consuming pattern. Each data batch necessitated distinct code implementations to cater to varying data formats or demands, resulting in redundancy that could be avoided. Drawing inspiration from the principles of Continuous

Integration/Continuous Deployment (CI/CD), we advocate for leveraging Object-Oriented Programming (OOP) as an apt solution. OOP, with its inherent characteristics of inheritance and polymorphism, offers a structured approach to tackling this issue.

The essence of OOP lies in encapsulating data and methods within objects, thereby safeguarding against unauthorized access or inadvertent modifications, thus ensuring data integrity, and minimizing errors. Inheritance facilitates the seamless integration of existing attributes/methods while accommodating the addition of new ones with minimal coding effort. Polymorphism enables the uniformity of function names and coding styles across different classes, thereby enhancing user experience.

By harnessing these OOP advantages, we propose a streamlined workflow consisting of three key steps. Firstly, the creation of an ADaM data object containing essential task data sets the foundation. Secondly, the selection of appropriate methods to process data within the object into a standardized format allows for user-friendly modifications. Finally, rendering the formatted layout into a document, such as a docx or pdf file, completes the process. This simplified workflow is not only efficient but also accessible to individuals with limited programming backgrounds due to its intuitive logic.

INTERNAL R PACKAGE DEVELOPMENT

In the realm of internal R package development, challenges often stem from inconsistencies in development environments and collaboration issues. Ensuring reproducibility and consistency in results across different environments is paramount. To address these concerns, we advocate for the adoption of Docker images as a comprehensive solution. A Docker image encapsulates not only the package itself but also all necessary dependencies, configurations, and versions. By creating a Docker image at the onset of package development, teams can essentially freeze the development environment, akin to taking a snapshot. This ensures that anyone utilizing the Docker image can reproduce the same results, regardless of their operating system, R language version, or package versions. Moreover, Docker facilitates multiple validations, as every user employs the same environment to test the code, thereby enhancing code reliability and reducing potential discrepancies.

Collaboration is another critical aspect of package development, and managing contributions from multiple team members can be challenging. Mistakes often arise during the distribution of work or when integrating changes from various contributors. To streamline collaboration and ensure version control, we advocate for the use of Git and GitHub. Git enables efficient code versioning, allowing developers to track changes, revert to previous versions if necessary, and merge contributions seamlessly. GitHub serves as a centralized platform for collaboration, providing features such as pull requests, issue tracking, and code review. By leveraging Git and GitHub, teams can maintain a stable version of their package for public use while continuing to develop new features or address existing issues in parallel. Furthermore, the transparency provided by GitHub allows for comprehensive tracking of changes, facilitating bug fixes, performance reviews, and overall project management.

INTERNAL R SHINY DEVELOPMENT

In the realm of R Shiny development, a common challenge arises from the proliferation of functions required to address various needs, resulting in intricate code structures that can be daunting to navigate. Our proposed solution involves simplifying the architecture by eschewing the creation of numerous functions. Instead, we advocate for a streamlined approach that focuses solely on the essential components of the server and UI aspects.

This minimalist approach is underpinned by the principles of Object-Oriented Programming (OOP), which are increasingly recognized for their ability to deliver simplicity and clarity. By embracing OOP within the Shiny development process, we alleviate the burden of contemplating complex functionalities. Developers can now direct their attention towards crafting a user interface that is both visually appealing and intuitive.

The key advantage of this approach lies in its efficiency: rather than reinventing the wheel with each new functionality requirement, developers can leverage pre-defined functions and methods encapsulated within OOP-style packages. This not only streamlines the development process but also ensures consistency and reliability across different components of the application.

In essence, our approach allows developers to focus on what truly matters: designing a seamless user experience. By abstracting away the complexities of functionality implementation through OOP, we empower developers to create Shiny applications that are both elegant and user-friendly, without sacrificing robustness or scalability.

CONCLUSION

Using open-sourced solutions in clinical trials for statistical analysis and reporting is an emerging trend. It allows us to access broader informative solutions. On one hand, people can learn other's experience and development and foster them into their own build. On the other hand, one might be flooded by extensive readings and cannot find a way to start with. This paper aims to help people in the pharmaceutical industry who want to build a team for open-sourced solutions. While we believe this trend can benefit the whole industry, the standardization and formalization, as we recognize, still needs a long way to go.

ACKNOWLEDGEMENT

We would like to appreciate the communication with Posit team for the detailed explanation for the product, and detailed advice from Phil Bowsher for current landscape.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Peng Zhang, Ph.D.
CIMS Global, LLC, Somerset, NJ, USA
pzhang@cims-global.com

Lizhong Liu, Ph.D.
CIMS Global, LLC, Somerset, NJ, USA
lliu@cims-global.com

Tai Xie, Ph.D.
CIMS Global, LLC, Somerset, NJ, USA
tomx@cims-global.com