PharmaSUG China 2025 - Paper SA-153

Ensure the Statistical Validity in the Context of open-source under GxP Compliant Environment

Frank Yang, CIMS Global;
Vivian Chang, CIMS Global;
Jade Lee, CIMS Global;
Peilin Zhou, CIMS Global
Peng Zhang, CIMS Global;

ABSTRACT

In recent years, R-based approach for clinical trials has rapidly gained attention in the pharmaceutical industry. There are many knowledge resources available, where one can efficiently incorporate those (e.g., {pharmaverse}, {openstatsware} and working groups) into their workflow. The available contribution will include CDISC standard data generation, static and interactive statistical results, and usage of LLM. Before proceeding on the real practice of automation or connection to LLM, one might ask about the validity or reliability of the package/statistical results, the extent on how we can trust such an approach. On the other hand, building a GxP complaint environment across the organization becomes an trending topic. In this session, we will introduce how to leverage open-sourced R package on internal development with good practice and validated evidence, and potential usage in pharmaceutical context (e.g., R Shiny, Quarto, LLM). We will also introduce our solution designed to help overcome these barriers and support compliant, R-based workflows in clinical trials.

INTRODUCTION

It is a common task that statistical programmers generate TFLs for statistical or clinical review, where it can happen in safety review, DMC meeting, clinical study report (CSR), etc. While one can review TFL instantly to have an idea of how current trial goes on, specific requirements such as CSR need the medical writer to come out report based on the results from TFLs. The inline table and figure usually take some effort to manually copy, type, or regenerate based on the values from TFLs. This process will introduce human error and can be time-consuming. In extreme case if there is any change from database, the regeneration of the report can take duplicated error. In this paper, we propose some alternative solutions that can streamline the process.

There are two types of solutions that one can build for:

- The first one is ‘one-click’ choice. By setting up all the options and parameters, when the data is finalized, one can just simply run the code and it will generate the expected results. This choice will usually use {quarto} or Rmarkdown format to process. After preparation of all the code, the solution can work for the updated data.
- The second one is ‘by-interaction’. Users will review the result and simultaneously edit their input and decide what to include. This method usually involves some tools with interaction such as {shiny} so that users can interact with the application, as there are some ‘variables’ during the process. For example, your summary for balanced or imbalanced results between your treatment can control group can be different. But note that once user finalizes the input from {shiny}, the finalized process can be similar to the first option as a render process with fixed parameters. Some similar open-source solutions such as {teal.reporter} [1] are using this approach to generate report from {teal} application for results review.

There is another concern from the users regarding the applications. People will worry about the validation. ‘What is your validation process’, or ‘Is your application validated?’ can be common questions during the conversation. Based on our practice, we may suggest separating the validation process into two parts, including statistical validity and operational qualification. The first part will ensure the values are correctly generated with evidence and the second part will ensure the application is connected to the

statistical package correctly. Validation reports (usually automated one) can be a preferred format for such proof.

In the following section, we will show how we build statistical validity, template-based narrative, application development and automation process.

STATISTICAL VALIDITY

In the CSR, TFLs will serve as reference or appendix and medical writers and add narratives based on the TFLs reviewed. To ensure the reliability of the process, we need to prove the results can be generated accordingly and correctly. Under these circumstances, before building the application, we propose to have a separate package dedicating for the TFLs generation. It is common that the organization has their own gallery of mock shells, and there has existed open-source solution of {tlg-catalog}[2] that one can refer or use as starting point. These templates can be generated using the {rtables} and {rlisting} packages (`table_shell()`), allowing users to simply input the corresponding variables from the ADaM dataset. By utilizing {flextable}, the user can document such template into preferred format.

GOOD DEVELOPMENT PRACTICE FOR PACKAGE

The planned list here provides a purpose of requirement, that what we expect the package to do. Once you have a planned shell list, the developer can prepare accordingly per requirements. It is recommended to have a good practice for R package development, so that the package will be in a good quality and easy to maintain in the future. Some organizations (e.g., {openstatsware}) have some nice guides [3] Good practices in R package development play a crucial role in ensuring the quality of the package, which, in turn, facilitates long-term maintenance of its features. A well-structured approach involves several key tools and techniques:

GitHub: For version control, collaboration, and code sharing.

- {roxygen2}: To write function headers and generate comprehensive documentation.
- {renv}: For managing the package environment and ensuring reproducibility.
- {devtools} and {usethis}: For streamlined package building and efficient development workflows.
- {testthat}: For implementing unit tests to validate the functionality of the package.
- {valtools} [4]: Generate Validation Report with requirement, test cases, test codes and results

A flow chart below in Figure 1 helps illustrate the idea

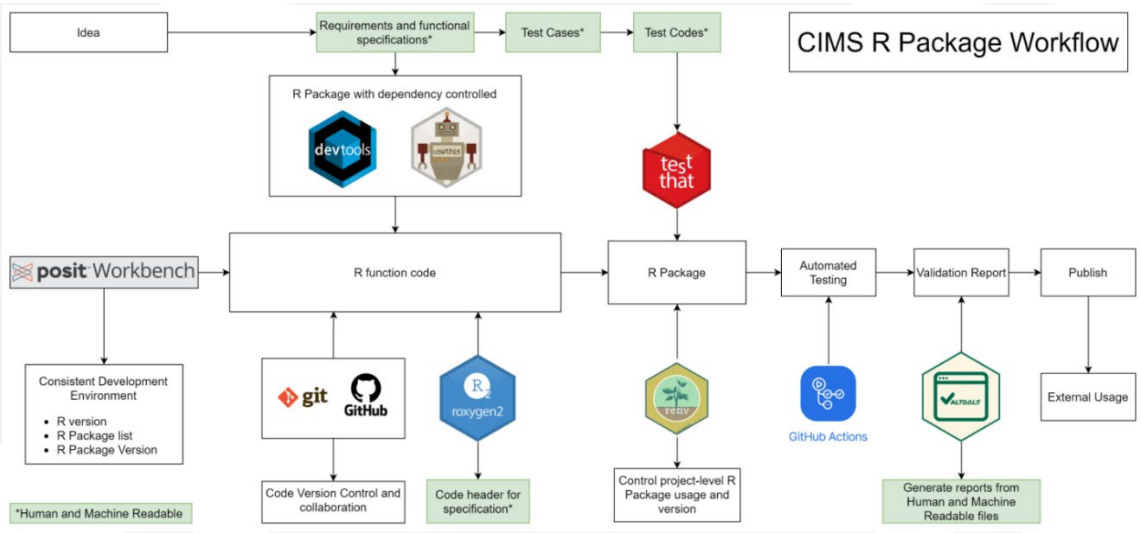

Figure 1. CIMS R Package Workflow

UNIT TESTING

As mentioned above, unit testing serves as an important role guaranteeing the code quality. For TFL results, it is common that the results are correctly generated by verifications from independent programmers. While there are different approaches about verification or QC process, we might suggest using ARD/ARS [5] generation of the dataset. Analysis Results Dataset (ARDS) is a standardized format to organize analysis results, such as tables and figures. ARDS helps to automate processes, reuse results, and validate data more efficiently. In the ARDS structure, all analysis values are placed in a single column, with each row corresponding to one value. Descriptive information, such as the name of the analysis variable and grouping details, is stored in separate columns. This approach makes it easier to validate and manage results. There are some public open-source packages about the topic such as {cards} [6] and {cardx} [7]

We wrote functions to convert the values in the table into the ARDS format. Descriptive information, such as the name of the SOC (System Organ Class) and Preferred Term (PT), is placed in the 'group1_level' and 'group2_level' columns. Statistical results are stored in the 'result' column, while statistical labels, such as counts (n') and percentages (pct'), are placed in the 'stat_name' column. After converting the table to a consistent standard data structure, the validation process can be efficient. An example of the shell and ARD structure are given in Figure 2 and Figure 3.

System Organ Class Preferred Term	Treatment Group (N=xx)	Control Group (N=xx)	Total (N=xx)
At least one AE	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Primary SOC 1	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Preferred Term 1	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Preferred Term 2	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Preferred Term 3	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Primary SOC 2	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Preferred Term 1	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Preferred Term 2	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)
Preferred Term 3	xxx (xx.x%)	xxx (xx.x%)	xxx (xx.x%)

Figure 2. Example of table shell for AE table

ARDS	group1	grouop1_level	group2	group2_level	avar_name	trt_var	trt	stat_name	result
						TRT01P	trtA	N	
						TRT01P	trtB	N	
						TRT01P	Total	N	
				At least one AE		TRT01P	trtA	n	
				At least one AE		TRT01P	trtA	pct	
				At least one AE		TRT01P	trtB	n	
				At least one AE		TRT01P	trtB	pct	
				At least one AE		TRT01P	Total	n	
				At least one AE		TRT01P	Total	pct	
	AESOC	<AESOC1>				TRT01P	trtA	n	
	AESOC	<AESOC1>				TRT01P	trtA	pct	
	AESOC	<AESOC1>				TRT01P	trtB	n	
	AESOC	<AESOC1>				TRT01P	trtB	pct	
	AESOC	<AESOC1>				TRT01P	Total	n	
	AESOC	<AESOC1>				TRT01P	Total	pct	
	AESOC	<AESOC1>	AEBODSYS	<AEBODSYS1>		TRT01P	trtA	n	
	AESOC	<AESOC1>	AEBODSYS	<AEBODSYS1>		TRT01P	trtA	pct	
	AESOC	<AESOC1>	AEBODSYS	<AEBODSYS1>		TRT01P	trtB	n	
	AESOC	<AESOC1>	AEBODSYS	<AEBODSYS1>		TRT01P	trtB	pct	
	AESOC	<AESOC1>	AEBODSYS	<AEBODSYS1>		TRT01P	Total	n	
	AESOC	<AESOC1>	AEBODSYS	<AEBODSYS1>		TRT01P	Total	pct	

Figure 3. Example of ARD structure for AE Table

VALIDATION REPORT

It is recommended to read PHUSE White paper [4] about potential validation method using {valtools} to establish corresponding evidence by providing requirements and testing plan. An example content is shown in Figure 4. After establishing the structure, your package will be validated with a report and the process can be automated in Github action as well. One well-written validation report using {valtools} will

be {drugdevelopR} and statistical calculation in clinical trial design and can be found online from Github [8]

Validation Report for cimstfl

Peng Zhang, Frank Yang, Nina Han, Vivian Chang, Jade Lee

2025-04-15

Contents

By combining those approaches together, these tools and best practices establish a robust foundation for developing high-quality, maintainable R packages. Once validation is completed with a validation report, the application can utilize such package with validated evidence.

CONNECTION TO OTHER PACKAGES

As the package achieve the statistical validity, one can use the package for other package or shiny apps to use. The statistical functions by giving parameters input make it intuitive to connect other functionality. Potential usage can include:

- Template-based narrative generation using ARD structure
- Summary report or session slides including intext table and narratives
- Interactive Shinyapps to display TFLs
- LLM using {shinychat} and {ellmer} to present correct results with traceability and validity

More details will be shared in our further development and papers.

PACKAGE MANAGEMENT

As {cimstfl} is developed as a package, we expect the data scientist in their daily work can use the package for their monitoring or shiny apps development. While during development, one can specify the package dependency in the DESCRIPTION file, it highly relies on the uncontrolled repository, which is usually public CRAN repository. Due to the nature of R, it will search the latest version of the package available from the repo and install it. We observe that there is a large difference before {rtables} 0.6.10 and afterwards due to the split of {rtables.officer}. In this case, the successful usage of {cimstfl} will be dependent on the version of {rtables} installed. To overcome this difficulty, one can follow the steps below to have an internal repository:

1. Focus on single R version first
2. Prepare a list of the package, including their dependent R package and system dependency
3. Conduct risk assessment and corresponding activity to ensure the package is selected on purpose [9]
4. Find a preferred snapshot and find the latest version of the package before that snapshot date. One can use public Posit Package Manager to achieve this goal.
5. Finalize the package with corresponding versions (one package one version) and system dependency.
6. Download the save the package into repo using Posit Package Manager or {miniCRAN} so that data scientist can utilize this repo

From this way, one can have their own internal repo and can guarantee the environment consistency

GXP COMPLIANT ENVIRONMENT

The GxP concept initiates lots of discussion of the open-source, where the organization expects the GxP Compliance for their environment to use. While the terms are different across organization's understanding, we might suggest to achieving the conditions below to at least maintain a minimum requirement of GxP compliance for the environment.

- Role-based access control
- Audit-trial for user's activity
- Controlled package management
- Validation of the environment setup per discussion with IT and QA

The organization can utilize Posit Workbench and Posit Package Manager to achieve the goal. Ensure fully utilizing the features from the licensed product will lead to better control. Alternative solution will include RStudio server with self-customized features, where needs more in-house expertise to achieve.

CONCLUSION

R provides a good format in terms of R package to ensure the standard can be efficiently used across different people. To fully utilize the feature, one should consider "making a good package" and "making a good place" for the package. With respect to the open-source, from the perspective of one organization, they should digest the information from open-source world to build their own controlled solution, where can help maintain the consistency and scalability. While open-source is usually free on the website, it takes other cost such as learning and technical implementation & validation to have a confident environment compared to other validated statistical computing language. Our paper suggests a potential solution based on our practice and would like to share with our audience the experience [10].

REFERENCES

[1] InsightsEngineering. “teal.reporter.” GitHub. n.d. Available at https://github.com/insightsengineering/teal.reporter.

[2] InsightsEngineering. “TLG Catalog.” InsightsEngineering GitHub Pages. n.d. Available at https://insightsengineering.github.io/tlg-catalog/stable/.

[3] OpenStatsWare. “OpenStatsWare Guide.” OpenStatsWare. n.d. Available at https://www.openstatsware.org/guide.html.

[4] PHUSE. “Data Visualisation & Open Source Technology – WP059.” PHUSE. n.d. Available at https://phuse.s3.eu-central-1.amazonaws.com/Deliverables/Data+Visualisation+%26+Open+Source+Technology/WP059.pdf.

[5] Clinical Data Interchange Standards Consortium (CDISC). “Analysis Results Standard (ARS).” CDISC. n.d. Available at https://www.cdisc.org/standards/foundational/analysis-results-standard.

[6] InsightsEngineering. “cards.” GitHub. n.d. Available at https://github.com/insightsengineering/cards.

[7] InsightsEngineering. “cardx.” GitHub. n.d. Available at https://github.com/insightsengineering/cardx.

[8] Sterniii3. “drugdevelopR Validation Report.” GitHub. n.d. Available at https://github.com/Sterniii3/drugdevelopR/blob/af99e25108d47b79bb0718b393ca90c9bbe188de/validation/validation.pdf.

[9] Cascone, Arianna D., and Chelsea N. Dickens. 2025. “R Package Risk Assessment: Balancing Programmatic and Human Inputs.” PHUSE US Connect Conference, Orlando, FL, March 17, 2025. Available at https://phuse.s3.eu-central-1.amazonaws.com/Archive/2025/Connect/US/Orlando/PAP_OS05.pdf

[10] Zhang, Peng, Jimmy Chen, Frank Yang, Vivian Chang, Jade Lee, and Tai Xie. 2025. “Generate Clinical Study Report (CSR) Document Using {quarto} and {shiny}.” PHUSE US Connect Conference, Orlando, FL, March 2025. Available at https://phuse.s3.eu-central-1.amazonaws.com/Archive/2025/Connect/US/Orlando/PAP_OS06.pdf.

CONTACT INFORMATION

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

Frank Yang
CIMS Global
fyang@cims-global.com