Paper OS06 – PHUSE US Connect 2025

Generate Clinical Study Report (CSR) document using {quarto} and {shiny}

Peng Zhang, CIMS Global, New Jersey, US
Jimmy Chen, CIMS Global, Taipei, Taiwan
Frank Yang, CIMS Global, Taipei, Taiwan
Vivian Chang, CIMS Global, Taipei, Taiwan
Jade Lee, CIMS Global, Taipei, Taiwan
Tai Xie, CIMS Global, New Jersey, US

ABSTRACT

In the pharmaceutical industry, TFLs can be generated through programming to support final clinical study report. However, by referring to the TFL results, medical writers need to manually transcribe or copy/paste the outputs into CSR. Such an entire process can consume several months depending on the complexity of the study and regulatory requirements. To overcome such difficulty, we would suggest two alternatives by using {quarto} or {shiny}, which is inclusive to derive description, narratives, inline tables, figures, etc., to derive CSR, through template or interactively. In this paper, we aim to address the availability of open-source solutions that can help the clinical reporting process. Similar use cases and examples will be provided as discussion.

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 a 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

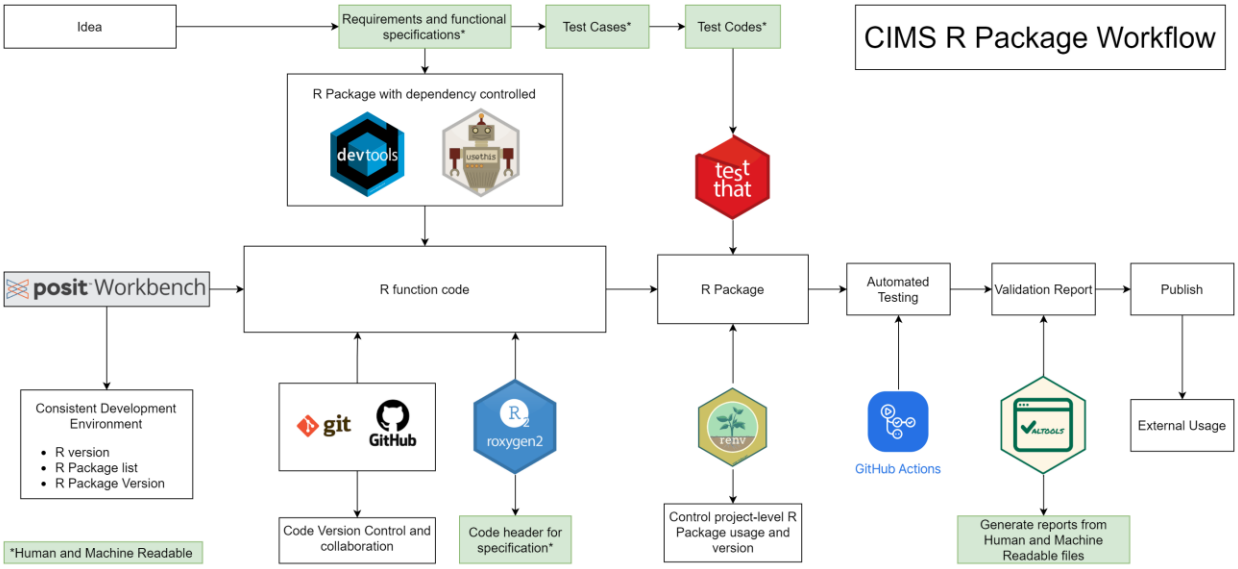

Figure 1: CIMS R Package Workflow

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

fyang

2025-01-24

Contents

Figure 4: An Example Validation Report Content

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.

TEMPLATED BASED CSR GENERATION

In medical writing for CSRs, there are often paragraphs across sections that follow the same structure. Because of this, creating templates for medical writers can help cut down on redundancy. These templates would include placeholders for variables—like statistics and values derived from the TFL (Tables, Figures, and Listings) packages. Once the necessary variables and values are gathered for each section, we can plug them into the templates, essentially "gluing" the data into the predefined structure, and quickly generate narratives for each section, ensuring consistency and accuracy across different sections of the CSR.

The package we developed consists of generating functions for narratives in CSR sections 9 to 11. Each function accepts an input of an ARDS structure, which is a transformation from TFLs and is constructed and validated by the statistical package mentioned above.

Depending on the type of data, the generating functions will tailor the template accordingly. The output includes two main components: one is a variable data frame that stores all necessary values and statistics under different variable names, and the other is a template containing placeholder variables. This template can be transferred and displayed without revealing the actual values. The overall narrative generation relies on a "gluing" process, where the variables containing statistics are integrated into the template. As a result, fully formed narratives are generated, which can then be used by medical writers.

For example, in section 9.1, we use the ARD transformed from the disposition table as an input.

group1	group1_level	trt_var	trt	stat_name	result
Study Completion Status	ONGOING	TRT01P	A: Drug X	num	2.400000e+01
Study Completion Status	ONGOING	TRT01P	A: Drug X	pct	1.791045e-01
Study Completion Status	ONGOING	TRT01P	B: Placebo	num	2.800000e+01
Study Completion Status	ONGOING	TRT01P	B: Placebo	pct	2.089552e-01
Study Completion Status	ONGOING	TRT01P	Total	num	5.200000e+01
Study Completion Status	ONGOING	TRT01P	Total	pct	1.940299e-01
Study Completion Status	DISCONTINUED	TRT01P	A: Drug X	num	4.200000e+01

Figure 5: The ARD of disposition table

The generation function of section 9.1 will take the ARD input and generate three components: var_df is the data frame that stores the values and its variable names, header is the header of the section, and md is the template with variable names as placeholders.

```
$var_df
         name      value
1  n_screened        268
2      n_rand        268
3   n_treated        268
4      n_comp        134
5    n_discon         82
6      n_ongo         52
7    name_gp1  A: Drug X
8      n_gp_1        134
9    name_gp2 B: Placebo
10     n_gp_2        134

$header
[1] "Disposition of Participants"

$md
[1] "A total of {n_screened} participants were initially screened for inclusion in the study. Of
these, {n_rand} participants were randomized into the study, and {n_treated} participants receive
d medication. The participants were divided into two groups: the {name_gp1} group ({n_gp_1}) and
the {name_gp2} group ({n_gp_2}). {n_comp} subjects had completed the study, while {n_discon} subj
ects discontinued prematurely, and {n_ongo} subjects were ongoing."
```

Figure 6. Three components of {stat2mw} generation function

By glueing [9] the values to the variable names, we can generate the narrative for this section as shown in Figure 7

```
Disposition of Participants

A total of 268 participants were initially screened for inclusion in the study. Of these, 268
participants were randomized into the study, and 268 participants received medication. The
participants were divided into two groups: the A: Drug X group (134) and the B: Placebo group
(134). 134 subjects had completed the study, while 82 subjects discontinued prematurely, and
52 subjects were ongoing.
```

Figure 7: Narratives of Dispositions of Patients

SHINY APPLICATION AND REPORT GENERATION

After you have validated statistical results and template-based approach, the next step is to connect all those features together to provide medical writers a working station to edit and combine the narratives together with in-line tables and figures.

As shown in Figure 8, the working station should include 4 parts:

- Output selection: This will enable users to include the necessary part that one wants to include. One issue in the generation of clinical study report is choosing the part of the results. Medical writer might need to copy-paste the results from TFLs as inline table in the report. In this way, medical writers can choose the desired part and save the section.
- Output review: By using statistical package, not only the results can be exported to docx or pdf, it can be also saved as machine readable format so that it can be directly displayed to the shiny page. One can utilize this panel as a review of the TFLs.
- Template edit: In this part, we provide a template for each section so that the automation process and

transform the ARD structure to the real number from variable names. The necessity of including two panels here is to avoid medical writer to delete variable names of numbers. In this panel, medical writer can only edit the description rather than the variable names, so the glue process can be correct.

- Narrative review: this panel will include the actual numbers/values from the TFLs and combine with the template.

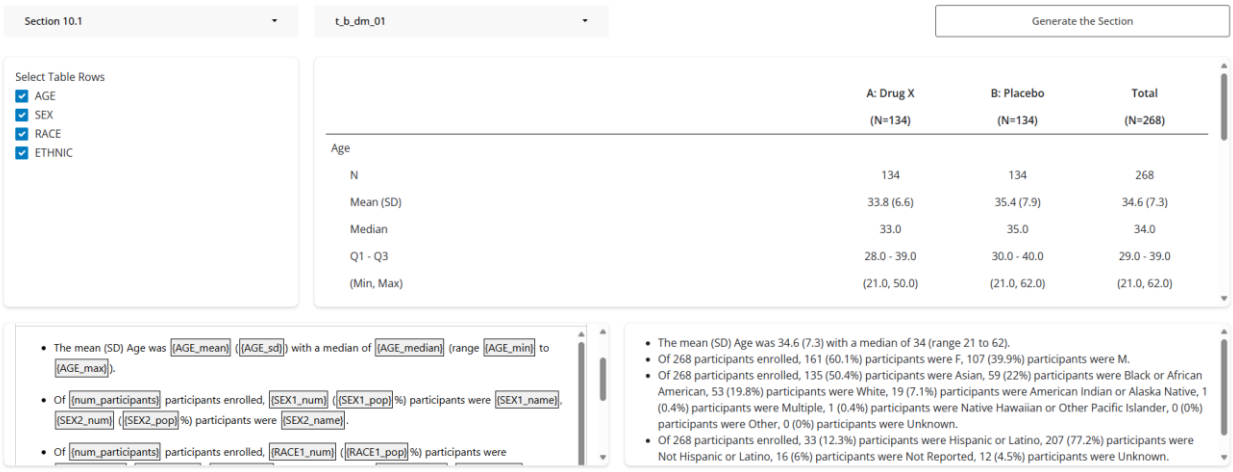

Figure 8 Example page for generating CSR using {shiny}

After medical writer reviews each section, the code and corresponding narratives will be saved into {quarto} files. The results will include the rendered outputs from such quarto files and the original quarto file for further replication purposes. An example rendered clinical study report will be shown in Figure 9. Note that for the report, if you have a pre-specified template from word document, it can be read into the documents directly using {quarto}

Clinical Study Report

9.1 Disposition of Participants

	A: Drug X (N=134)	B: Placebo (N=134)	Total (N=268)
Number of Patients Screened			268
Number of Randomized Subjects	134 (100%)	134 (100%)	268 (100%)
Number of Treated Subjects	134 (100%)	134 (100%)	268 (100%)
Study Completion Status			
COMPLETED	68 (50.7%)	66 (49.3%)	134 (50.0%)
ONGOING	24 (17.9%)	28 (20.9%)	52 (19.4%)
DISCONTINUED	42 (31.3%)	40 (29.9%)	82 (30.6%)
ADVERSE EVENT	3 (2.2%)	6 (4.5%)	9 (3.4%)
DEATH	25 (18.7%)	23 (17.2%)	48 (17.9%)
LACK OF EFFICACY	2 (1.5%)	2 (1.5%)	4 (1.5%)
PHYSICIAN DECISION	2 (1.5%)	3 (2.2%)	5 (1.9%)
PROTOCOL VIOLATION	5 (3.7%)	3 (2.2%)	8 (3%)
WITHDRAWAL BY PARENT/GUARDIAN	4 (3%)	2 (1.5%)	6 (2.2%)
WITHDRAWAL BY SUBJECT	1 (0.7%)	1 (0.7%)	2 (0.7%)
Treatment Completion Status			
COMPLETED	68 (50.7%)	66 (49.3%)	134 (50.0%)
ONGOING	24 (17.9%)	28 (20.9%)	52 (19.4%)
DISCONTINUED	42 (31.3%)	40 (29.9%)	82 (30.6%)

A total of 268 participants were initially screened for inclusion in the study. Of these, 268 participants were randomized into the study, and 268 participants received medication. The participants were divided into two groups: the A: Drug X group (134) and the B: Placebo group (134). 134 subjects had completed the study, while 82 subjects discontinued prematurely, and 52 subjects were ongoing.

10.2.1 Population at Screening

	A: Drug X (N=134)	B: Placebo (N=134)	Total (N=268)
Age			
N	134	134	268
Mean (SD)	33.8 (6.6)	35.4 (7.9)	34.6 (7.3)
Median	33.0	35.0	34.0
Q1 - Q3	28.0 - 39.0	30.0 - 40.0	29.0 - 39.0
(Min, Max)	(21.0, 50.0)	(21.0, 62.0)	(21.0, 62.0)
Sex			
N	134	134	268
F	79 (59.0%)	82 (61.2%)	161 (60.1%)
M	55 (41.0%)	52 (38.8%)	107 (39.9%)
Race			
N	134	134	268
ASIAN	68 (50.7%)	67 (50.0%)	135 (50.4%)
BLACK OR AFRICAN AMERICAN	31 (23.1%)	28 (20.9%)	59 (22.0%)
WHITE	27 (20.1%)	26 (19.4%)	53 (19.8%)
AMERICAN INDIAN OR ALASKA NATIVE	8 (6.0%)	11 (8.2%)	19 (7.1%)
MULTIPLE	0	1 (0.7%)	1 (0.4%)
NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER	0	1 (0.7%)	1 (0.4%)
OTHER	0	0	0
UNKNOWN	0	0	0
Ethnicity			
N	134	134	268
HISPANIC OR LATINO	15 (11.2%)	18 (13.4%)	33 (12.3%)
NOT HISPANIC OR LATINO	104 (77.6%)	103 (76.9%)	207 (77.2%)

Figure 9 An example report generating CSR by using {quarto}

APPLICATION VALIDATION

For shiny apps validation, one might need to provide operational qualification (OQ) procedure to serve as evidence. In this case, one should provide validation plan per requirement and corresponding test cases, so that the application is validated per requirements that each requirement is working. A manual screenshot process or utilizing {shinytest2} for automation process can be acceptable. The process is usually documented in SOP and complaint to the defined procedure and should be discussed with QA department ahead.

DEPLOYMENT AND AUTOMOATION OF AVAILABLE SHINY APPS

Deploying a document as a Shiny application is an effective way to enhance sharing and enable personalized interaction with the document. Once deployed, users with the appropriate permissions can access the content at any time, and all users see the same version of the content. This contrasts with sharing a file, where each user might end up with different versions, leading to inconsistencies.

However, deploying a Shiny application often requires specialized professionals to validate and configure the application using IQ documents before automation can be implemented. This process is typically time-consuming and repetitive. Moreover, IT professionals often have other projects and responsibilities, which limit the time and frequency they can dedicate to deploying Shiny applications. These constraints are necessary to ensure the quality of the deployment but can restrict the flexibility and development timeline of the application.

Even with thorough documentation and verification of manual processes, human error can never be entirely eliminated. Automating the deployment of Shiny applications addresses these challenges by reducing manual effort, minimizing the risk of human error, and increasing the deployment frequency. This allows applications to reach users more quickly, enabling faster feedback and further improvements, ultimately enhancing the development cycle and

user experience.

GitHub
GitHub is a web-based platform primarily used for version control and collaboration in software development. It allows developers to host, review, and manage code, making teamwork and coding projects more efficient.

Pull Request
A pull request is a feature in GitHub (and other version control platforms) that allows developers to propose changes to a codebase by submitting their work for review before it is merged into the main branch. It facilitates collaboration by enabling team members to review, discuss, and suggest improvements to the code. Pull requests help enhance code quality by encouraging peer reviews, identifying potential bugs, ensuring adherence to coding standards, and fostering knowledge sharing within the team. This process ensures that only a well-reviewed, high-quality code is integrated into the project.

GitHub Actions
GitHub Actions is a powerful automation tool within GitHub that allows developers to create workflows for building, testing, and deploying code directly from their repositories. It is highly customizable and uses YAML configuration files to define tasks, called "actions," that run automatically based on specific triggers, such as pushing code, creating pull requests, or setting scheduled events. With GitHub Actions, teams can streamline CI/CD pipelines, automate repetitive tasks, and ensure faster, more reliable software delivery, all within the GitHub ecosystem.

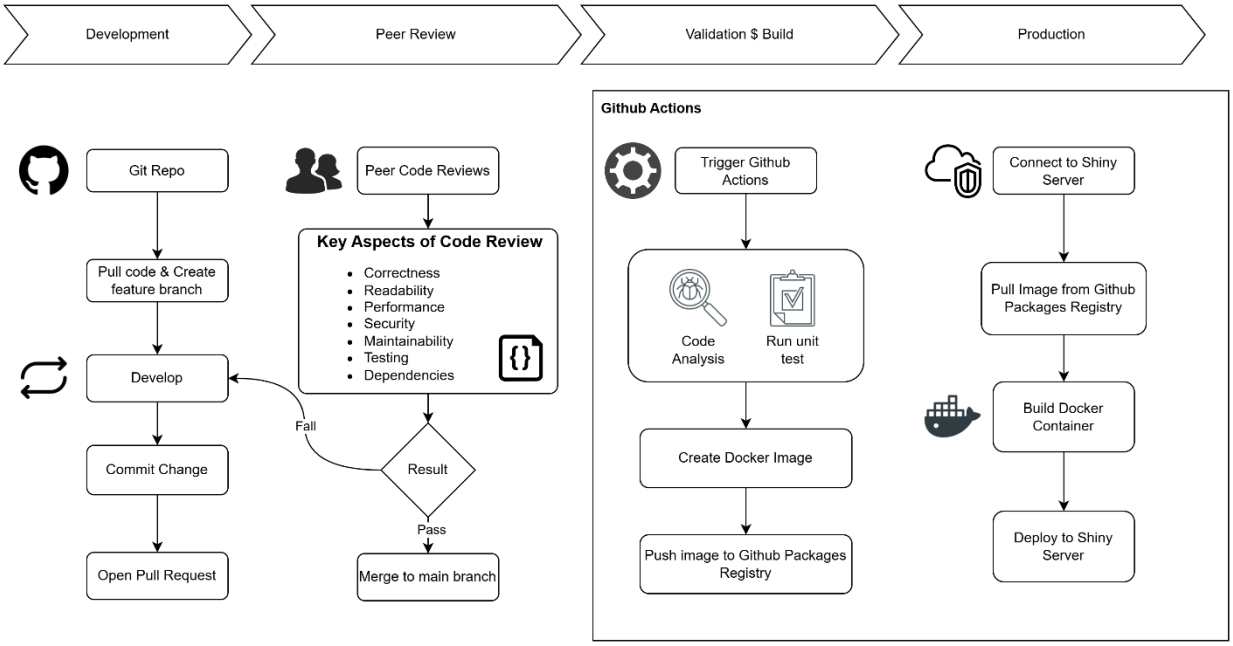

Figure 10 Flow Chart for Github Actions

There are many tools available for automating deployment, but we chose GitHub due to its reliability, widespread adoption, and robust feature set. GitHub not only allows us to host and manage our code but also provides tools like GitHub Actions to streamline automated deployment. With GitHub, we maintain a complete history of code modifications, which cannot be altered, making it invaluable for auditing purposes.

To ensure that each submitted feature is stable and reliable, we utilize the pull request mechanism. This process ensures that all code undergoes thorough review and discussion, guaranteeing it meets functional requirements while adhering to security and performance standards.

When a feature is approved and merged into the main branch, it triggers GitHub Actions to execute predefined automation workflows, which are defined in YAML files for traceability and easy adjustments. The automation process starts with a quality check of the code, using both static analysis (to detect issues like syntax errors, security vulnerabilities, and style violations without executing the code) and dynamic analysis (to monitor runtime behavior and performance). Automated test scripts are also executed to verify code logic and application flows. If any test fails, the deployment process halts, and the test results are immediately reported to the development team.

Once the code passes all checks, the next step is to build and package it into a Docker image, which is then pushed

to the GitHub Packages Registry. Using a secure connection between GitHub and our local Shiny server, GitHub Actions initiates the deployment process by pulling the latest Docker image to the server. The deployment process completes with the necessary build and parameterization steps, ensuring a seamless release of the Shiny application.

DISCUSSION

In this paper, we propose a way of using R packages and shiny applications to provide a working station for medical writers to review the results and edit the narratives, following with automated reports by rendering process. By providing robust development processes and validation evidence, a reliable process can be streamlines to generate the CSR report especially for Section 9 to 11.

While there are lots of AI available solutions that can help generate paragraphs based on the TFLs by reading those results, we believe that the CSR process, as the final step to the regulatory, should be rigorous and fully validated. While people always have concern about the correctness about the results from AI, we believe such template approach with statistical evidence can be feasible to ensure reliability and flexibility as well in the process.

REFERENCES

[1] https://github.com/insightsengineering/teal.reporter

[2] https://insightsengineering.github.io/tlg-catalog/stable/

[3] https://www.openstatsware.org/guide.html

[4] https://phuse.s3.eu-central-1.amazonaws.com/Deliverables/Data+Visualisation+%26+Open+Source+Technology/WP059.pdf

[5] https://www.cdisc.org/standards/foundational/analysis-results-standard

[6] https://github.com/insightsengineering/cards

[7] https://github.com/insightsengineering/cardx

[8] https://github.com/Sterniii3/drugdevelopR/blob/af99e25108d47b79bb0718b393ca90c9bbe188de/validation/validation.pdf

[9] https://glue.tidyverse.org/

CONTACT INFORMATION (HEADER 1)

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

Author Name: Peng Zhang
Company: CIMS Global
Address: 285 Davidson Ave, Suite 504, Somerset, NJ, 08873, US
Email: pzhang@cims-global.com