PharmaSUG 2021 - Paper SI-084

A Process to Validate Internal Developed R Package under Regulatory Environment

Madhusudhan Ginnaram, Simiao Ye, Yalin Zhu, Yilong Zhang, Merck & Co., Inc., Kenilworth, NJ, USA.

Amin Shirazi, Iowa State University

ABSTRACT

The use of R in regulated environments has been on the rise and is playing a role in the submission process of clinical trial analysis results to regulatory agencies. Validation is an important part of R package development and is an essential step in the regulated environment. In this paper we propose a detailed and practical validation procedure for testing R packages under regulatory environment especially for Analysis and Reporting (A&R) deliverables in clinical trials. In general, R package testing involves 3 major stages: planning, implementation and reporting. Our proposal follows the same principles that can enhance the traceability and robustness of an R package development within an organization. The proposed procedure helps an organization to develop an R package following a risk-based approach to assess R package within a validated infrastructure (https://www.pharmar.org/white-paper/) suggested by R Validation Hub. We use a recently developed R package `r2rtf` (https://github.com/Merck/r2rtf) to illustrate the proposed procedures for R package validation. We hope this proposed validation procedure can provide much needed guidance on utilizing R with adequate testing for clinical data A&R within an organization and eventual submission to regulatory agencies.

INTRODUCTION

In Analysis and Reporting (A&R) of clinical trials, testing or validation is an important step in a software development lifecycle (SDLC) to build high quality software and meet regulatory requirements. In R package development, testing also plays an important role. There are several widely used R packages (e.g. devtools(https://github.com/r-lib/devtools), testthat((https://github.com/r-lib/testthat/), usethis (https://github.com/r-lib/usethis)) and tools (e.g. github action, Travis CI, Jenkins) to simplify and automate the testing workflow in developing an R package.

A general SDLC process for R package development has recently been discussed in Zhu et al. (2020). An R Package helps in developing, maintaining and sharing R code, with a standardized structure. The SDLC process described by Zhu et al. is applicable to the regulated environment of A&R in clinical trials. In general, the A&R process involves 4 stages (define, develop, validate and operations).

In developing a robust R package, proper testing shall be considered and implemented in every stage of the SDLC. Basically, R package testing involves 3 major stages: planning, implementation and reporting. Our proposal follows the same principles that can enhance the traceability and robustness of an R package development within an organization. The proposed procedure helps an organization to develop an R package following a risk-based approach and assess R package within a validated infrastructure (https://www.pharmar.org/white-paper/) suggested by R Validation Hub.

In this paper, we propose a detailed process to leverage existing tools to complete the validation of an R package developed within an organization for regulatory deliverables. We use a recently developed R package `r2rtf` (https://github.com/Merck/r2rtf) to illustrate the proposed procedures for testing or validation. We hope this proposed validation procedure can provide much needed guidance on utilizing R with adequate testing for clinical data Analysis and Reporting within an organization and eventual submission to regulatory agencies.

PLANNING TEST CASES

Planning testing cases is critical to build formal and robust validation of an R function. The general expectation is to have at least one testing case for each function in an R package. The testing cases

shall align with the potential use cases for a function. The testing plan for each function shall be initiated at the planning stage by the project lead and reviewed by stakeholders right after the function specification is defined. The initial testing cases may not cover all the required scenarios during planning stage but can be expanded and updated after team starts to develop the functions.

We illustrate the idea by using the `rtf_title` function in `r2rtf` package. The testing plan of `rtf_title` can be developed using `testthat` R package. For example, the code below indicates we shall test the format of the title for the `rtf_title` function.

```
test_that("title format", {})
```

All the testing cases are saved in the `tests/testthat` folder. The general workflow of using `testthat` in R environment has been discussed by Wickham (2015). With a pre-defined function specification, the project lead in concert with other stakeholders shall define the level of testing required for each function and provide description of the testing scope based on the specification of each function. For example, the type of testing can be classified as

- developer testing: developers writing testing code by themselves.
- independent testing: another developer writing testing code by referring the source code
- double programming: another developer writing testing code by referring the function specification without referring the source code.

It is recommended to use developer testing only for simple or standardized functions. Independent testing or double programming is recommended for functions that are complex or critical for A&R purposes. The testing environment can be setup using the `usethis::use_testthat()` function.
It is recommended to use `testthat::use_test()` function to streamline and automate the setup of testing files. We suggest using pre-fixed filename to classify the types of testing with the same file name for the source code in the R folder, the project lead developer will set up the corresponding test files in a consistent filename structure.

In r2rtf package, there are 32 functions developed. We illustrate the idea using `rtf_title`, a function to control the features for table and figure title saved in an RTF file. The function structure for `rtf_title` is shown below. The meaning of each argument can be found in
https://merck.github.io/r2rtf/reference/rtf_title.html

```
rtf_title <- function(tbl,
                      title = NULL,
                      subtitle = NULL,
                      text_font = 1,
                      text_format = NULL,
                      text_font_size = 12,
                      text_color = NULL,
                      text_background_color = NULL,
                      text_justification = "c",
                      text_indent_first = 0,
                      text_indent_left = 0,
                      text_indent_right = 0,
                      text_space = 1,
                      text_space_before = 180,
                      text_space_after = 180,
                      text_convert = TRUE)
```

To create testing cases, below are code examples for different validation strategy in an R package folder structure:

```
usethis::use_test("developer-testing-rtf_title")
usethis::use_test("independent-testing-rtf_title")
usethis::use_test("double-programming-rtf_title")
```

The testing files in the planning stage only contain meta information to describe the expectation of each testing case. It will enhance team communication between developer and validator. The project lead developer is expected to draft the testing cases in plain language saved in the desc argument of the `testthat::test_that` function. Below is an example of a testing plan for the `rtf_title` function in `r2rtf` package:

```
test_that("input argument checks ", {})
test_that("title format ", {})
test_that("title font color and background color", {})
test_that("title justification, spacing and indentation ", {})
test_that("multiple subtitles", {})
```

Generally, we suggest creating one R file for every function being developed as shown in the Figure 1 below. If multiple functions are developed within a single R file under R folder, then multiple testing files with corresponding functions names can be created in `testthat` folder.

Figure 1: Screenshot from Rstudio

At the end of defining stage, the project lead will summarize the list of functions into an excel file or a report. The project lead in discussion with other stakeholders will review the level of testing needed for each function in the package. An example excel file (Table 1) is as below if `rtf_title` is classified as independent review.

Program Name	Description	Program Validation Category	Developer	Status	Double Programmer	Status	Independent Reviewer	Status
rtf_title	Function to add title attributes to the table	Independent testing	Madhu	In Progress	N/A	N/A	Simiao	In Progress

Table 1: Function Name with Initial Status

IMPLEMENTING TEST CASES

After the current round of development is completed for an R package, developers assigned to validate those functions will start to implement testing cases based on the assignment by project lead. In principle, developer and validator of a function must be different for independent testing and double programming. If the function involves complicated algorithm, double programming is necessary to ensure accuracy of the

results in different cases. For each testing file, it is suggested to assign only one developer to implement testing cases. The validator will complete the testing files based on the information in the function specifications and testing plan. It is recommended to use variants of `expect_xxx()` functions available in the `testthat` package to complete validation. For example: partial testing code of `rtf_title` is as below. More details of using these `expect_xxx()` functions can be found in (Wickham 2015).

```
test_that("input argument checks", {
  expect_error(rtf_title(tbl_1, title = 1))
  expect_error(rtf_title(tbl_1, subtitle = NA))
  expect_error(rtf_title(tbl_1, text_font = "a"))
  expect_error(rtf_title(tbl_1, text_font_size = "b"))
  expect_error(rtf_title(tbl_1, text_format = "z"))
})
```

In testing complicated object such as figure, rtf file or html file, it is recommended to use the snapshot feature in the `testthat` package (https://testthat.r-lib.org/articles/snapshotting.html).
The snapshot feature requires the tester to manually approve any change of the output and ensure proper documentations of the testing results to enhance traceability. For example, we use code below to take a snapshot of the file for testing default margin of the file after manual review.

```
test_that("RTF page, margin encoding", {
  data1 <- iris[1:2,]
  data2 <- data1 %>% rtf_body(as_colheader = TRUE,
                              col_rel_width = c(5,9,13,18,9) )
  data3 <- data2 %>%  rtf_encode_table()
  expect_snapshot_output(data3)
})
```

In testing tables, listings or figures (TLFs) in RTF file, it is also recommended to separate the testing of analysis results and table format. For analysis results, the final dataset can be saved in an `.Rdata` file. The validator can load the saved data and compare it with the testing results. This helps in improving the efficiency of testing by concentrating testing efforts on the actual data contained in the TLFs rather than testing the final output which would contain various styling attributes.

A validator is also expected to execute testing case and provide feedback to developer accordingly. The tests in `r2rtf` package were run by individual testers during implementation. The “Run Tests” button in Rstudio or `devtools::test_file()` help validator to check the testing results for a single file.
The resulting errors or warnings were reported in the RStudio Build panel as shown in Figure 2. The failure of an individual unit test resulted in display of error message that described the context of failure, which helped in locating and fixing the bugs.

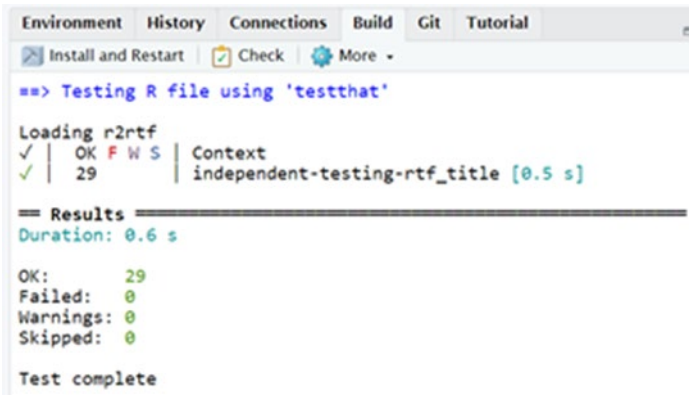

Figure 2: Test results of rtf_title

`devtools::test()` function can be used to automatically run all the testing cases in R package. Project lead can use this function to review overall status of the testing results and shown in Figure 3.

```
> devtools::test()
Loading r2rtf
Testing r2rtf
✓ |  OK F W S | Context
✓ |   3       | test-convert_latex [0.1 s]
✓ |   8       | Independent testing for as_rtf_colheader.R [0.4 s]
✓ |   8       | Independent testing for as_rtf_header.R [0.6 s]
✓ |   0     1 | Development testing for as_rtf_table.R
```

Figure 3: r2rtf overall testing status

The validator should strive to have an informative name and cover a single unit of functionality for each unit test. The validator should also create informative message so that the source of the problem can be identified efficiently. After the test is completed the validation tracker shall be updated accordingly as shown below Table 2

Program Name	Description	Program Validation Category	Developer	Status	Double Programmer	Status	Independent Reviewer	Status
rtf_title	Function to add data title attributes to the table	Independent testing	Madhu	Completed	N/A	N/A	Simiao	Completed

Table 2: Function Name with Final Status

REPORTING FOR TEST CASES

Code coverage summarize the percentage of source code being executed by the testing cases. It is a useful metric to review the implemented testing cases. `covr` package or coverage functions available in `devtools` package can be used to ascertain the level of testing coverage. Validator can run the `covr::file_coverage()` or `covr::report()` to check the level of testing coverage for individual functions. The project lead would be responsible for checking the code coverage of a R package using the `covr::package_coverage()`. It is recommended that the functions or package being tested have approximately 80% coverage. This type of testing coverage report also helps reviewers identify the level of validation performed in the R package in a human readable format.

Internally, we ran the `covr::file_coverage()` or `covr::report()` to check the testing coverage for their developed functions. For example, the code coverage for `rtf_title` function is as below Figure 4.

```
> covr::file_coverage("~/r2rtf/R/rtf_title.R",
+                     "~/r2rtf/tests/testthat/test-independent-testing-rtf_title.R")
Test passed
Test passed
Test passed
Test passed
Test passed
Coverage: 49.02%
~/r2rtf/R/rtf_title.R: 49.02%
```

Figure 4: File Coverage of rtf_title function

The overall code coverage for the r2rtf package can also be generated after running the `covr::package_coverage()` as shown below Figure 5. This step was followed by the project lead throughout the package development. This ensured identification in testing gaps for individual functions and incremental increase in testing effectiveness.

Tests and Code Coverage Report

Overall coverage - 96.04%

Show 10 entries Search:

File	Lines	Relevant	Covered	Missed	Hits / Line	Coverage
R/match_arg.R	89	26	21	5	867	80.77%
R/as_rtf_table.R	124	45	38	7	14	84.44%
R/content_create.R	357	86	73	13	14	84.88%
R/rtf_encode_table.R	216	99	90	9	8	90.91%
R/rtf_body.R	256	68	63	5	68	92.65%
R/rtf_encode_figure.R	132	54	52	2	3	96.30%
R/rtf_page.R	237	69	67	2	63	97.10%
R/rtf_source.R	157	47	46	1	13	97.87%
R/rtf_footnote.R	226	47	46	1	18	97.87%
R/rtf_colheader.R	178	52	51	1	92	98.08%

Showing 1 to 10 of 30 entries Previous 1 2 3 Next

Figure 5: r2rtf Overall code Coverage

The reporting of testing coverage of `r2rtf` can also be found in codecov website (https://codecov.io/gh/Merck/r2rtf/tree/master/R) with example below Figure 6.

Files	≡	•	•	•	Coverage
as_rtf_pageby.R	115	113	0	2	98.26%
as_rtf_paragraph.R	20	20	0	0	100.00%
as_rtf_table.R	45	38	0	7	84.44%
check_args.R	19	19	0	0	100.00%
content_create.R	86	73	0	13	84.88%

Figure 6: r2rtf package in codecov website

DISCUSSION

In the A&R process, proper validation of an internal developed R package is critical to ensure regulatory compliance and leads to high quality deliverables. We proposed an easy-to-follow process by leveraging existing R package testing frameworks to complete the three stages (e.g. planning, implementation and reporting) for R package validation within an organization. The process is illustrated by a recently developed R package, `r2rtf`.

While the development involves multiple developers, we recommended the developer team follows agile management approach. Version control is also a critical part during the development and validation of an internal developed R package.

Although the proposed workflow covers the major stages of R package testing, we acknowledge the implementation of the workflow might vary widely. We hope this proposed validation procedure can provide much needed guidance on utilizing R with adequate testing for clinical data Analysis and Reporting within an organization and eventual submission to regulatory agencies.

REFERENCES

Zhu Yalin, Jajoo Rinki, Bai Clare, Nepal Sarad, Woodie Daniel, Anderson Keaven, Zhang Yilong. “R Package Oriented Software Development Life Cycle in Regulated Clinical Trial Environments” PHUSE US Connect (2020)

Wickham, Hadley. *R packages: organize, test, document, and share your code*. " O'Reilly Media, Inc.", 2015.

ACKNOWLEDGEMENTS

The authors would like to thank management teams from Merck & Co., Inc., Kenilworth, NJ, USA, for their advice on this paper/presentation and want to thank Uday Preetham Palukuru from Merck & Co., Inc., Kenilworth, NJ, USA for valuable inputs on the paper.

CONTACT INFORMATION

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

Madhusudhan Ginnaram
Merck & Co., Inc., Kenilworth, NJ, USA
e-mail: madhusudhan.ginnaram.reddy@merck.com

Yilong Zhang, Ph.D.
Merck & Co., Inc., Kenilworth, NJ, USA
e-mail: yilong.zhang@merck.com