PharmaSUG 2022 - Paper AD-076

Exploring R as a validation tool for statistical programming in clinical trials

Uday Preetham Palukuru, Runcheng Li, Nileshkumar Patel, Changhong Shi
Merck & Co., Inc., Kenilworth, NJ, USA

ABSTRACT

SAS® has long been used as the go to and in some cases the only software for statistical programming to support clinical trials analysis and reporting (A&R). In recent years, R has emerged as an alternative software for use in clinical trial A&R. However, the widespread use of R is still hindered by some practical considerations such as, learning curve for SAS® only users and limited knowledge on the capabilities of R for data analysis.

In this paper, we will discuss the unique strengths of R in the validation of clinical trial A&R deliverables. Exploratory results from the efforts to validate SAS® generated outputs using R will be discussed. We mainly focus on the validation of the widely used deliverables in clinical trial data analysis i.e. Tables, Listings and Figures (TLFs). We will showcase the interactive features available in R that help in validation of figures We will also demonstrate the validation capability of R by comparing mockup table package and final output tables, so any discrepancy in titles, subtitles or footnotes can be identified efficiently.

The ability of R to perform complicated statistical procedures with reduced programming code complexity and easier execution will also be discussed in this paper. By integrating R shiny applications into the validation workflow, the improvement in efficiency of validation due to the easy-to-use interface, is also discussed. We hope the discussed features and capabilities will help promote the use of R as an efficient validation tool in clinical trial A&R.

INTRODUCTION

CLINICAL TRIALS ANALYSIS AND REPORTING

Analysis and Reporting (A&R) is a critical process in clinical trial data analysis. It requires careful planning to generate all A&R deliverables and typically follows a clearly defined rigorous process to ensure high quality. The typical A&R programming SDLC performed in clinical trials can be divided into four phases: define, develop, validate and operation. *Figure 1* provides an overview of these 4 phases.

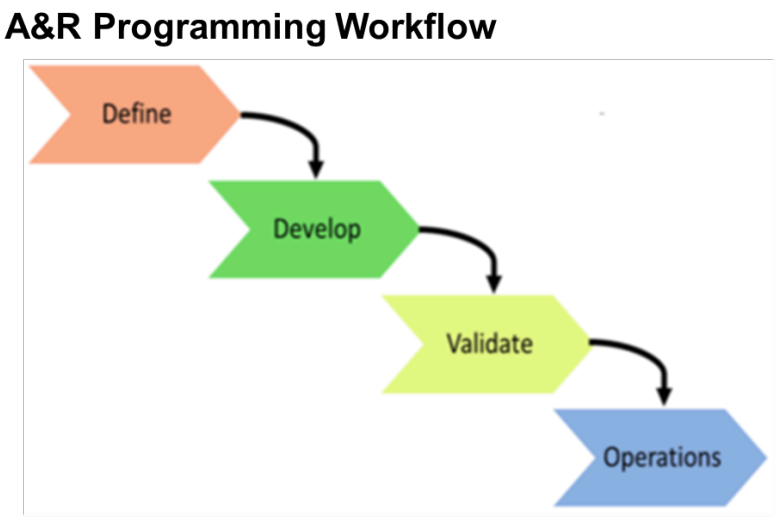


Figure 1. The 4 phases of the A&R programming workflow in clinical trials.

The define phase is a critical phase in the clinical trial development lifecycle as the requirement specifications for deliverables and resources needed to generate the deliverables are gathered and documented. The lead programmer and statistician discuss the programming and analysis requirements for the planned deliverables with the programming team. The validation plan to ensure the quality of individual deliverables is also planned and documented at this stage. After the specifications are developed and individual deliverables tasks are assigned, the programming development occurs in the develop phase. The lead programmer, in consultation with the clinical trial team, obtains data as needed for program development.

In the validate phase, programs developed in the develop phase are validated according to the validation plan and requirement specifications. The programs are validated to promote quality and regulatory compliance. The lead programmer in consultation with lead statistician reviews completed deliverables and conducts a test run of deliverables to ensure quality.

In the operations phase, the validated programs are executed with production data to generate the final version of required deliverables. Necessary updates are done in this phase to address new requirements or issues. The lead programmer and statistician review the production deliverables and communicate the availability of deliverables and completion of programming activities. Requestors communicate any required changes with the lead programmer. Specifications/programs are updated as required (Nepal, et al., 2021).

SAS® USAGE IN CLINICAL TRIALS ANALYSIS

SAS® is an integrated software system that is utilized in many industries to perform tasks like data management, reporting, statistical analysis, and application development. SAS® consists of Base SAS software along with an Integrated Development Environment (IDE) that enables users to write code using the graphical user interface (GUI) or in the editor. The Base SAS software consists of different modules such as DATA steps, SAS macro facility, Base SAS procedures and Output Delivery System to enable users to seamlessly import and analyze data as well as create output reports. Many additional modules are available that can be added to enable additional functionality based on an organization's requirements (Li, 2013).

SAS® has seen widespread use as part of clinical trials A&R ever since the introduction of the PH-Clinical component in the late 1990s (Wang, 1996). The use of SAS® in clinical trials was further accelerated by the acknowledgement of FDA as acceptable analysis software for new drug applications (NDA) in 2002.

The typical usage of SAS® in clinical trials analysis and reporting (A&R) is illustrated in *Figure 2*. The generation of Clinical Data Interchange Standards Consortium (CDISC) datasets i.e., Study Data Tabulation Model (SDTM) and Analysis Data Model (ADaM) datasets, along with generation of various submission ready reports form the bulk of day-to-day operations using SAS® in clinical trials analysis. The development structure followed in most organizations employing SAS® for clinical trial data analysis is also shown.

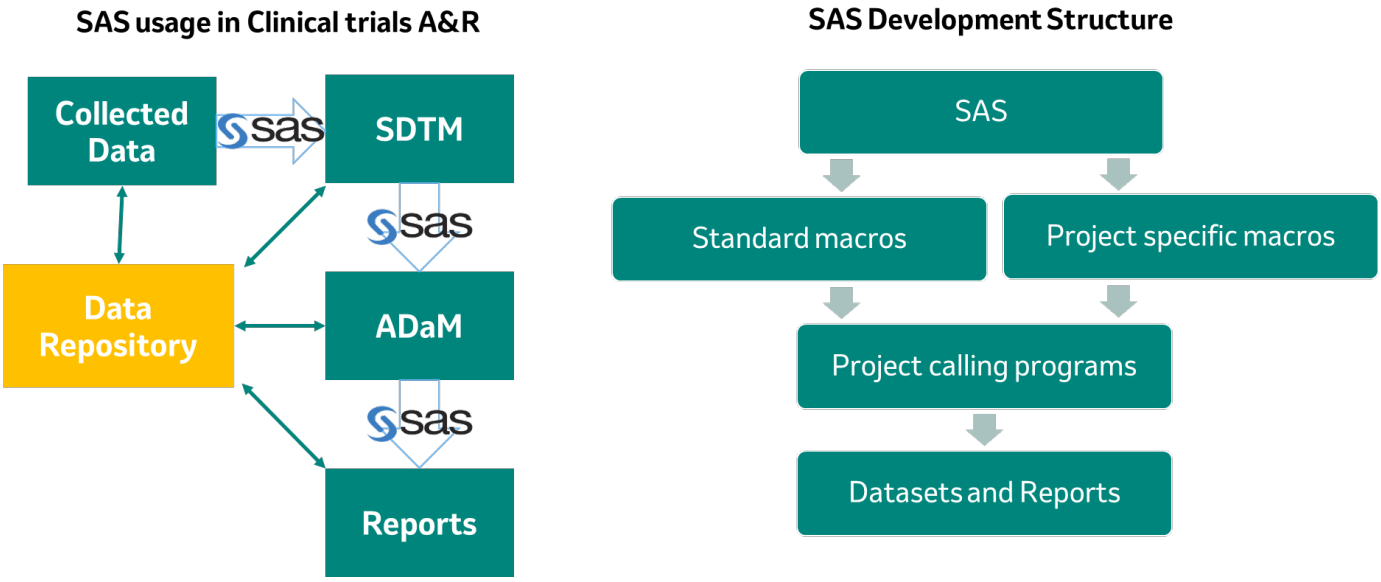


Figure 2. Usage and Development Structure of SAS® in clinical trials analysis and reporting.

R USAGE IN CLINICAL TRIALS

R is an open-source programming language with an integrated suite of software facilities used for statistical computing, data processing and graphics needs. R supports object-oriented programming along with procedural programming with the use of a command-line interpreter and functions. An R function is a set of R statements organized together for a specific task and can have arguments. An R package is a collection of R functions that enables users to expand R functionality by offering data wrangling, statistical, graphical, and reporting techniques. Most R packages are user-created and are hosted on online repositories as open source. These packages can be installed and customized by users in a plug and play fashion according to project requirements. Highly interactive IDE provided by vendors such as RStudio can be added to base R software for code development and deployment (Wickham & Grolemund, 2016).

R and its ecosystem have seen widespread use in recent years within the pharmaceutical industry for planning and analyzing data in different stages of drug development. With the release of guidance from R Foundation as well as cross-industry initiatives such as R Validation Hub and Transcelerate regarding using of R in regulated clinical trial environments, there has been an uptick in the number of organizations employing R in clinical trial A&R. R is being increasingly used for generation of submission ready deliverables with the usage and development structure depicted in *Figure 3*. The development structure with R also mirrors that with SAS® due to the highly regulated environment of clinical trial A&R. There are several ongoing efforts to submit clinical data analyzed using R for regulatory approval of NDAs.

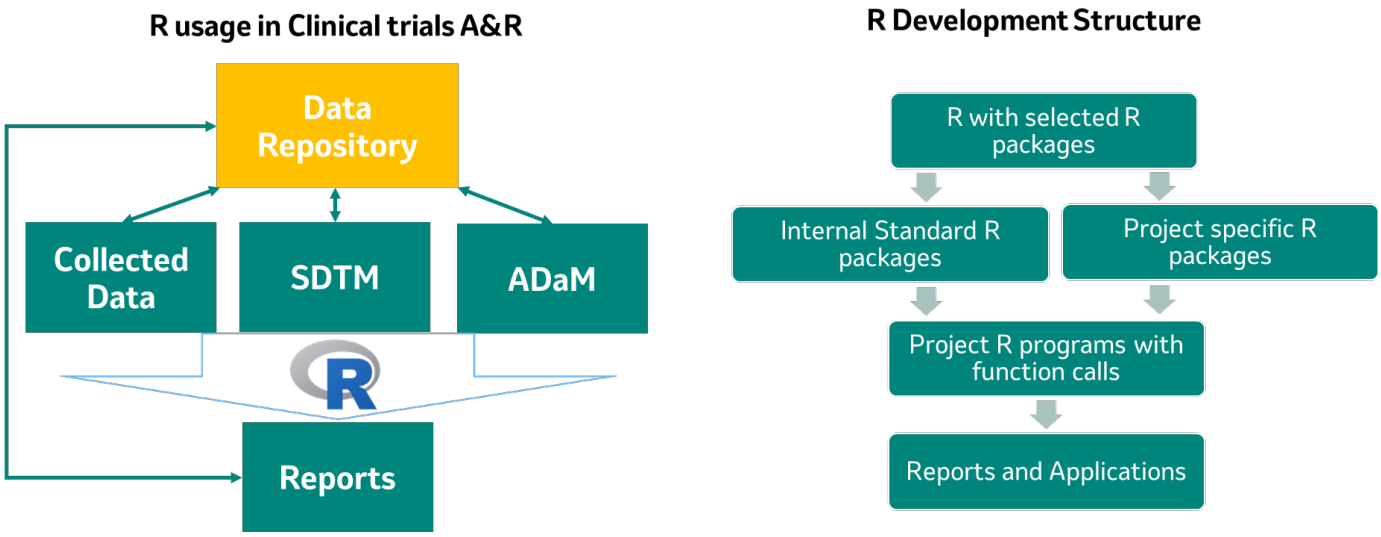


Figure 3. Usage and Development Structure of R in clinical trials analysis and reporting.

MOTIVATION

Due to the highly regulated nature of clinical trials, the validation of outputs is necessary. Validation is also necessary to ensure the quality of the clinical reports. The validation of generated deliverables is incorporated as part of standard operating procedures such as in the validate phase of the A&R programming workflow. Although SAS® and R are being used in clinical trial environments for A&R, they are used in a standalone capacity. There has been a dearth of literature reporting the use of SAS® and R concurrently in clinical trial A&R. Through concurrent use, the individual strengths of SAS® and R as summarized in *Table 1* can be leveraged to increase programming efficiency as well as increase the confidence in generated deliverables.

Leveraging Individual Strengths

SAS	R
Large number of existing validated procedures for A&R use	Large number of open-source packages for A&R use
Established SOPs for use in clinical trial A&R due to widespread industry use	Highly interactive and customizable data visualization packages
Support Infrastructure and commercial validation in place due to widespread use	Ability to customize features based on user needs, owing to being open source
	Availability of novel statistical methods

Table 1. Summary of individual strengths of SAS® and R.

In this paper we explore the use of R as a validation tool to validate deliverables produced by SAS®. We discuss 2 use cases where R is used as validation tool for validating figures and tables. We also explore using R to validate complex statistical procedures and integrating R shiny applications to improve

efficiency of validation. With these use cases we hope to demonstrate the capabilities of R and help promote its implementation as an efficient validation tool in clinical trial data analysis.

R AS A VALIDATION TOOL

FIGURES COMPARISON USE CASE

Figures are routinely generated as part of deliverables during the clinical trial A&R process. Figures often contain plots or graphs of information contained in complementary tables with accompanying text formatted according to specifications. To validate figures, a validation programmer typically uses the PROC COMPARE in SAS® to check the output datasets generated by the production programmer and manually compares the production output with specifications. However, figure creation is more complex than creating a table. Additional considerations, such as axis-scale, color, symbol, range, also need to be checked. Changes in any of these parameters can cause the plots or graphs to look different than what is expected but cannot be ascertained using SAS® PROC COMPARE.

Many different R packages enable automatic figure checks by reading in the generated output in entirety rather than just the associated dataset. We explored the feasibility of using these R packages for validation purposes by using a method to extract the figures from output .rtf files and comparing the pixel values within the figures. One of the R packages used for image processing is the *imager* which is based on *CImg*, a C++ library (Tschumperlé, Tilmant, & Barra, 2004). *CImg* provides an easy-to-use and consistent application programming interface (API) for image processing, which *imager* largely replicates. *CImg* supports images rendered in up to four dimensions, which makes it suitable for basic video processing/hyperspectral imaging as well (Barthelmé, 2021). With the *imager* package, pixel values can be easily read from a plot or graph and compared to a reference plot or graph. To simplify working with *imager* it is recommended to use image file formats such as .png or .jpg. Since figures generated within the A&R lifecycle are not typically stored as image files, we also developed a workflow to extract the plot or graph embedded in .rtf file as an image file. The *imager* package is then used to highlight the discrepancies between the generated and reference files. The main activities in the workflow are discussed below:

Extract Data from .rtf file

Plots or graphs in .rtf files are stored as binary code as seen below:

```
\pard{\par}{\*\bkmkstart IDX}{\*\bkmkend IDX}\pard\plain\qc{{\*\shppict{\pict\pngblip\picwgoal9000\pichgoal6754
89504E470D0A1A0A0000000D49484452000004E2000003AA0802000000BE486D9E00000009704859730000 1EC200001EC2016ED0753E000020004944415478 9C
ECDD7D741BD59DF8FF9BC479822486 6D435B905A6DDB6D9312164280143780815068 17F761E94281D2DAB53849F71424C7D242D978251FDB0758EBD4327B76E3
6C02688142811E2881028140080 90EB84044EDD3A4A52569356529D0EE7160096449FCFBE37EB9BFE9E8C1B2340F77A4F7EB2F4973255F6946D67CE673EFE74E
9B9898100000000000E861BAD71D0000000000E0FF47980A00000000D008612A00000000402384A900000000008D10A602000000003442980A00000000D00861
2A00000000402384A900000000008D10A602000000003442980A00000000D008612A00000000402384A900000000008D10A602000000003442980A00000000D0
08612A00000000402384A900000000008D10A602000000003442980A00000000D008612A00000000402384A900000000008D10A602000000003442980A000000
00D04883D71D8083BEFFFDEF3FF4D0438B172F5EBE7CB9D77D0100000005017F6EDDB376DDAB45B6FBDB5E2579836131616387A0955028F4BBDFFDCEEB5E0000
0000A83BD5449A64536BD9D1471FFDBBDFFD2E140AB5B6B67ADD170000000075E1A9A79E7AEAA9A7AA7905C2D45AD6D8D828840885428944C2EBBE00000000A8
0B1313135586A9945002000000000688430B52C0F3FFCF0B469D3A64D9B96BFE997BFFCE5B4922EBCF0C2FC67DD7DF7DDF92D4F3FFD74E7DF0A00000000688D30
```

The .rtf file is read into R by using readLines function and the binary code related to plot or graph is isolated and converted to byte list. A snippet of the code that enables such a conversion is shown below:

Code snippet to extract binary data from plots/graphs

```
extract_rtf_png <- function(rtfFile){
  #read in RTF
  myrtf <- readLines(rtfFile)

  blist <- c()
  ind <- 0
  for (str in myrtf){
     if (ind){
       if (length(grep("\\}\\}\\}",str))){
         break
       }
       #split string 2 character
       sst <- strsplit(str, "")[[1]]
```

```
      list <- paste0(sst[c(TRUE, FALSE)], sst[c(FALSE, TRUE)])
      blist <- c(blist,list)
    }

    if (length(grep("pict",str))){
      ind <- 1
    }
  }
  bytelist <- as.raw(as.hexmode(blist))
  return(bytelist)
}
```

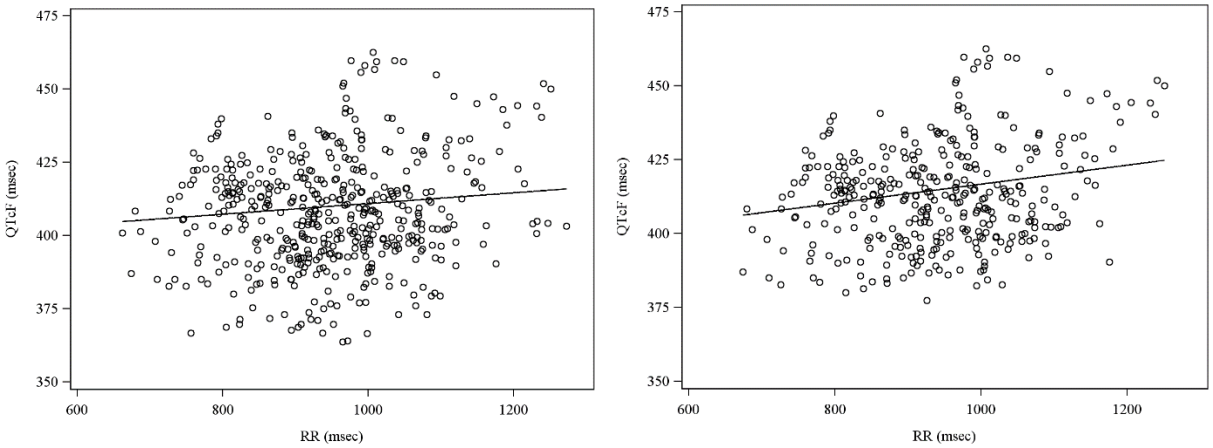
Once the byte list is extracted, the writeBin function can be used to convert the byte value back to .png file or .jpg file format, which can be used as the input for *imager* package.

Read in image data and compare by pixel

This step is simplified owing to the powerful image processing functions available within the *imager* package to process the images from .png files. The function load.image() can be used to read in images. The as.pixset(plotA - plotB) function declaration can be used to return the difference between two pixset matrices. The differences can be highlighted by superimposing the differences onto the compared pixset matrix as seen in the bottom image in *Figure 4*. A snippet of code that does the comparison is shown below:

```
diff <- as.pixset(plotA - plotB)
 comp <- colorise(plotA, diff, "red", alpha=1)
```

Input (Original scatter plots on left and reference scatter plot on right):



Output (Original plot with differences highlighted in red):

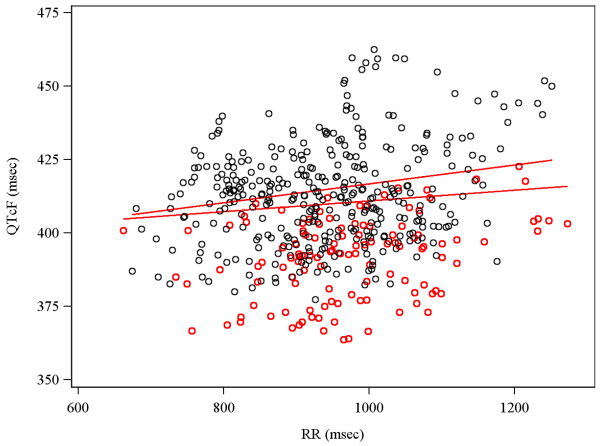


Figure 4. Inputs and resulting comparison output of imager package.

As can be seen from the output image in *Figure 4*, the discrepancies between two images are highlighted in red which is enables easier validation of graphical outputs. This workflow uses less resource than a comparable SAS® program to compare the differences. Thus, R can be used as a validation tool to increase efficiency and to automate the validation process.

TITLES AND FOOTNOTES IN TABLES COMPARISON USE CASE

Tables are the most commonly generated deliverables as part of the clinical study reports (CSR). They summarize clinical data and display various statistics to help a reviewer interpret the data. The programmer generates these tables based on the statistician's specifications in the form of comprehensive mockup shells. These mock up shells include required titles and footnotes for each output. The titles and footnotes can be a major source of programming inefficiency and errors. They are usually added to the program code manually, and the wording might change frequently during many review cycles. Automating comparisons between generated tables and mockup shells using R can mitigate this programming inefficiency as well as increase the quality of generated outputs. We explored this possibility. The features in the workflow that enable extraction of the title and footnote from both .docx file (mockup) and .rtf file (output tables) and display the discrepancies programmatically are discussed below:

Extract titles and footnotes data from .rtf file

Rich Text Format (.rtf) is a proprietary document file format with published specifications developed by Microsoft®. The .rtf file can be input into R by using the function readLines(). Then the title and footnote can be extracted from the input list. A snippet of code used to extract title and footnote is shown below:

Code snippet to extract rtf title and footnote

```
extract_rtf_title_footnote <- function(Filepath){
  lstall <- list()
  listrtf<-list.files(Filepath,"\\.rtf$")
  for (rtf in listrtf){
    #read in RTF
    myrtf <- readLines(file.path(Filepath, rtf))

    title <- c()
    end_note <- c()
    trigger <- 0

    for (txt in myrtf){
      txt_trim <- str_trim(txt)
```

```
    if (txt_trim != ""){
      # extract the title text
      if (length(grep("\\\\",str_sub(txt_trim,1,1))) == 0 && length(grep("\\{",str_sub(txt_trim,1,1))) == 0){
        if (sum(txt==title) == 0){
          title <- append(title,txt)
        }
      }
      #extract the end notes
      if (length(grep("\\\\intbl\\\\ql\\\\sb30\\\\sa30", txt_trim))){
        trigger <- 1
      }
      if (trigger == 1 && length(grep("\\\\fi-160\\\\li160\\\\ri1", txt_trim))){
        pos <- unlist(gregexpr("\\\\fi-160\\\\li160\\\\ri1", txt_trim)) + 19
        note <- str_sub(txt, pos)
        if (sum(note==end_note) == 0){
          end_note <- append(end_note,note)
        }
      }
    }
  }
  extract <- list(filename=rtf, title=title, footnote=end_note)
  lstall <- append(lstall, list(extract))
 }
 return(lstall)
}
```

Extract titles and footnotes data from .docx

DOCX is Word Open XML Format Document, with more complex encoding than .rtf, but it is used more often for documentation purposes. There are some R packages available such as *officer* to extract plain text from .docx files. However, these packages usually import all the text in the document without any filter. The package user must identify the required content from the input list object. This is especially inefficient when considering variation in mockup titles and footnotes between authors. One approach to mitigate this issue is the use of standardized mockup shell. A snippet of code used to extract title and footnote from a standardized mockup is shown below:

Code snippet to extract .docx title and footnote

```
#extract_mockup
extract_mockup <- function(mockup){
 #read in mockup
 doc <- read_docx(mockup)
 summ <- docx_summary(doc)

 #extract title
 title <- c()
 sub <- summ[summ$content_type == "paragraph" & summ$text != "",]

 type <-c()
 num <- c()
 tit <- c()
 for (text in sub$text){
   if (str_sub(text,1,3) == 'Tab' | str_sub(text,1,3) == 'Fig'){
     splt <- strsplit(text, " ")
     type <- append(type, unlist(splt)[1])
     num <- append(num, unlist(splt)[2])
     tit <- append(tit, paste(unlist(splt)[c(3:length(unlist(splt)))], collapse = ' '))
```

```
  }
 }

 #extract footnote
 sub_foot <- summ[summ$content_type == "table cell" & summ$cell_id == 1 & summ$text != "",]
 footer <- sub_foot %>% group_by(doc_index) %>% slice(which.max(row_id))
 ft <- footer$text

 extract <- list(type=type, num=num, title=tit, footnote=ft)
 return(extract)
}
```

Output tables and mockup comparison

Once all tables and footnotes from both generated tables and mockup are extracted, they can be compared to check for discrepancies. A simple approach is to use the title to match the output table with that in mockup. However, each mockup table may have replicates in the output, with differing subtitles, based on populations used. To maximize the chance of correctly matching the generated table and mockup, one can add special characters in the mockup to represent changeable content in both title and footnote. In the example shown below we use the '$' character. Consider that one adverse event table is present in the mockup, and output tables are replicated based on different sub-setting criteria. This scenario can be handled using the logic and code snippet as shown below, which matches, each output table with its corresponding mockup.

Code snippet to match output tables with standard mockup

```
mock <- 'Participants With $ Adverse Events'
table1 <- 'Participants With Drug-related Adverse Events'
#mockup
m1 <- strsplit(mock, '\\s+')[[1]]
m1 <- m1[m1 != '$']
#table title
t1 <- strsplit(table1, '\\s+')[[1]]
m1 %in% t1
```

After the matching is complete, footnote checking can be automated by calculating the generalized Levenshtein distance between two-character values. This returns the minimal possibly weighted number of insertions, deletions and substitutions needed to transform one string to another. The higher the value, the more discrepancies between two strings. A .csv report can be generated based on the comparison, as shown in *Figure 5*. Column C in the report informs the reviewer if the output table exists in the mockup, and column D displays the footnote distance with a 0, indicating they are an exact match. The footnote displayed in table3.rtf file is:

- Output table: "*Based on participants with non-missing values at both baseline and Week 12"*
- Mockup: "*Based on participants with non-missing values at both baseline and Week 12."*

The difference between the 2 footnotes is the period symbol '.', which is difficult to spot during a manual check, however, this can be identified by the code automatically.

	A	B	C	D
1		FileName	Exist in Mokcup (Y/N)	Footnote (distance)
2	1	table1.rtf	Y	0
3	2	table2.rtf	N	NA
4	3	table3.rtf	Y	1

Figure 5. A .csv report of the comparisons between output tables and standard mockup.

COMPLEX STATISTICAL PROCEDURES VALIDATION USING R

Statistical analysis of clinical trial data forms the core of A&R. The statistical methods used to analyze the data can be as simple as counts of subjects with a specified condition, to as complex as, using non-proportional hazard to analyze overall survival rate. The programming code necessary to transform the input data to a format that can be analyzed using inbuilt SAS® procedures can be quite complex and resource intensive. Various methods can be employed by organizations to reduce resource use and promote wider use of these complex statistical procedures, including development of standardized macros. By employing standard macros, the complex code needed to analyze the data can be managed at a central location by lead or standards programmers, while study programmers can use the macros to generate multiple outputs via plug and play methodology. This enables efficient resource management as well timely delivery of outputs for review.

In most organizations, the validation programmers assigned to validate the outputs of such complex statistical procedures often use the standard macros to check the results of the production programmer. A major drawback of this approach is the quality of the validation could suffer if the standard macros used for validation contains unknown bugs that can affect the generated statistical results. This risk can only be mitigated if the validation programmer writes separate code to perform steps achieved in the standard macro used by the production programmer. In the case of complex statistical procedures, it is unreasonable to expect the validation programmer to develop their own code as it is time consuming.

A possible solution can be to use R to validate these complex statistical procedures. Many R packages such as *tidyverse* reduce the amount of code needed to transform input data into analysis ready format. Similarly, many R packages such as *survival* and survRM2 are capable of performing overall survival rate analysis using non-proportional hazard. The use of these packages in concert with graphical packages such as *ggplot2* enables the validation programmer to validate the results and outputs of the production programmer independently. This approach also increases the confidence in the validation from a reviewer's perspective as the statistical results are truly software independent and are based solely on the input data.

This approach of using R as a validation tool for complex statistical analysis can be further enhanced by using *Shiny* package. R *Shiny* package enables users to integrate interactive web applications within their program code to enhance user interface and increase useability. Further validation of complex statistical procedures can be achieved using *testthat* package. R *testthat* package offers many features for validation of programming code including snapshot testing. A framework incorporating *testthat* package has been developed and demonstrated for validation in regulatory environment (Ginnaram, Ye, Shirazi, & Zhang, 2021). The snapshot testing feature of *testthat* package can be readily integrated into existing validation workflow and can help automate validation process. The automation of validation process in concert with use of test coverage reporting features available in *testthat* package would further increase programming efficiency and confidence in analysis results.

CONCLUSION

Both SAS® and R provide tools to analyze clinical trial data and are presently in use by various organizations. Although SAS® has been the traditional choice for many years, the popularity of R as an open-source software platform is leading to more use cases. There are many published results indicating the strengths of an individual software versus the other, but a dearth of literature exploring concurrent use of both. In this paper we explored the use of R as a validation tool and its concurrent use with SAS® in increasing the efficiency of implementing the A&R process. The cases of using R in validation of deliverables mainly figures and tables showcase the flexibility of R. The use of R to validate results of complex statistical procedures in SAS® increases the confidence in the validation of the results. Given that R packages are open-source software, it is important to ensure that the packages used for validation are themselves qualified and validated. Since validation programs are not required as part of submission package to regulatory authorities, implementation of R as a validation tool would not hinder an organization's existing submission procedures.

REFERENCES

Barthelmé, S. (2021, January 29). imager: an R package for image processing. Retrieved March 11, 2022, from https://dahtah.github.io/imager/

Ginnaram, M., Ye, S., Shirazi, A., & Zhang, Y. (2021). A Process to Validate Internal Developed R Package under Regulatory Environment. PharmaSUG, SI-084. Retrieved March 11, 2022, from https://www.pharmasug.org/proceedings/2021/SI/PharmaSUG-2021-SI-084.pdf

Li, A. (2013). Handbook of SAS® DATA Step Programming. Boca Raton: CRC Press.

Nepal, S., Palukuru, U. P., Wu, P., Ginnaram, M., Patel, R., Chimbirithy, A. V., . . . Zhang, Y. (2021). Agile Project Management in Analysis and Reporting of Late Stage Clinical Trials. PharmaSUG, SI-083. Retrieved March 11, 2022, from https://www.pharmasug.org/proceedings/2021/SI/PharmaSUG-2021-SI-083.pdf

Tschumperlé, D., Tilmant, C., & Barra, V. (2004, October 01). The CImg Library. French Institute for Research in Computer Science and Automation. Retrieved March 11, 2022, from http://cimg.eu/index.html

Wang, M. (1996). The Role of SAS Programmers in Clinical Trial Data Analysis. NESUG 96 Proceedings. Industry Applications, pp. 330-335. Boston: NESUG. Retrieved March 15, 2022, from https://www.lexjansen.com/nesug/nesug96/NESUG96061.pdf

Wickham, H., & Grolemund, G. (2016). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. Sebastopol: O'Reilly Media. Retrieved March 15, 2022, from https://r4ds.had.co.nz/

ACKNOWLEDGMENTS

The authors would like to thank BAAMR and management teams from Merck & Co., Inc., Kenilworth, NJ, USA, for their advice on this paper/presentation.

CONTACT INFORMATION

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

Uday Preetham Palukuru, Ph.D.
Merck & Co., Inc., Kenilworth, NJ, USA
E-mail: preetham.palukuru@merck.com

Runcheng Li
Merck & Co., Inc., Kenilworth, NJ, USA
E-mail: runcheng.li@merck.com

Nileshkumar Patel
Merck & Co., Inc., Kenilworth, NJ, USA
E-mail: nileshkumar.patel@merck.com

Changhong Shi
Merck & Co., Inc., Kenilworth, NJ, USA
E-mail: changhong_shi@merck.com