PharmaSUG China 2023 - Paper AP-104

For SAS programmers, it's time to learn R!

Sookie Kong, Sanofi

ABSTRACT

As pharmaceutical industry begins to embrace open-source languages like R, the need to replicate typical analysis and reporting tasks using R is crucial. We all know that learning a new language can be challenging. Therefore, we aim to make the transition to R as effortless as possible by providing concrete examples relevant to daily programming. This paper covers the differences between SAS syntax and R syntax in clinical trial programming, which can help SAS-based peers quickly master the use of R methods and gain a competitive edge in the industry.

INTRODUCTION

Open-source software has increasingly become a popular choice in the pharmaceutical industry due to its simplicity and free advantages. R, in particular, has emerged as the most attractive software as it is not only a required topic for statistics but is also widely used in the academic community. R's clear instructions, ease of use for free and open source, and recent popularity in the industry have made it a favored choice for clinical statistical programming. Multinational pharmaceuticals and regulators are exploring and attempting to use the new R language written programs to make submissions, marking a shift from submissions that were mainly based on the SAS language in the past.

The R Consortium recently announced that on Nov 22nd, 2021, the R Submissions Working Group, which includes members from over 10 pharmaceutical companies and regulatory agencies, successfully submitted an R-based test submission package through the FDA eCTD gateway. Additionally, CDISC has established an open-source tool group, the CDISC Open Source Alliance (COSA), which is no longer limited to SAS. Many repositories are officially recognized by COSA as open-source projects focused on implementing or developing CDISC standards to drive innovation in the CDISC community, with most of them being R-based, such as Admiral, defineR, tfrmt, and Tplyr.

We have reason to believe that the extensive usage of R for supplying indications is accessible and that SAS programmers in the industry should prioritize learning R. This paper will guide SAS programmers with CDISC backgrounds on how to easily apply R.

WHY USING R

FOR SPONSOR:

1. R is more cost-effective than SAS as it is open source and does not require a license. Companies can save a significant amount of money by opting for R instead of paying for proprietary software.
2. R has faster adaptability to new trends due to its open-source nature, allowing anyone to contribute to package development, read source code, and resolve issues. When dealing with messy and unstructured datasets, R may be a better option as it offers more flexible and innovative methods for analysis.

FOR SUPERVISORY AUTHORITIES:

1. The reason why FDA submissions are mainly based on SAS is that it is a validated software. In contrast, open-source software is not always perceived as validated, and there may be concerns about support. However, the R user community is vast, and users can seek help and resolve issues quickly.
2. When comparing the merits of SAS and R within the industry, we often overlook the fact that SAS has been used for decades, and we know precisely what it does, how it does it, and how quickly it can do it. This is not the same for R because it is not yet as widely used. It is recommended that sponsors consult with the review teams to ensure the choice and suitability of statistical software packages. As

R gains popularity, experience will be gained, and its capabilities will be more objectively assessed.

CHALLENGES AND OPPORTUNITIES FOR PROGRAMMERS:

SAS programmers face several challenges as they consider transitioning to open-source programming languages. One of the main obstacles is the lack of experience in using open-source languages, which can make it difficult to learn a new language quickly. Moreover, many companies do not provide sufficient training to help SAS programmers learn R programming language, which is necessary for them to apply it to actual work projects.

However, the transition to open-source programming presents a wealth of opportunities for SAS programmers. By mastering the R programming language, they can gain access to more innovative and advanced data science technologies, which can help them become top talents in the industry. In addition, mastering R can also help SAS programmers work across industries, making it easier for them to find new and exciting career opportunities. With the growing popularity of open-source programming, SAS programmers who are willing to invest time and effort into learning R can greatly enhance their career prospects and become leaders in their field.

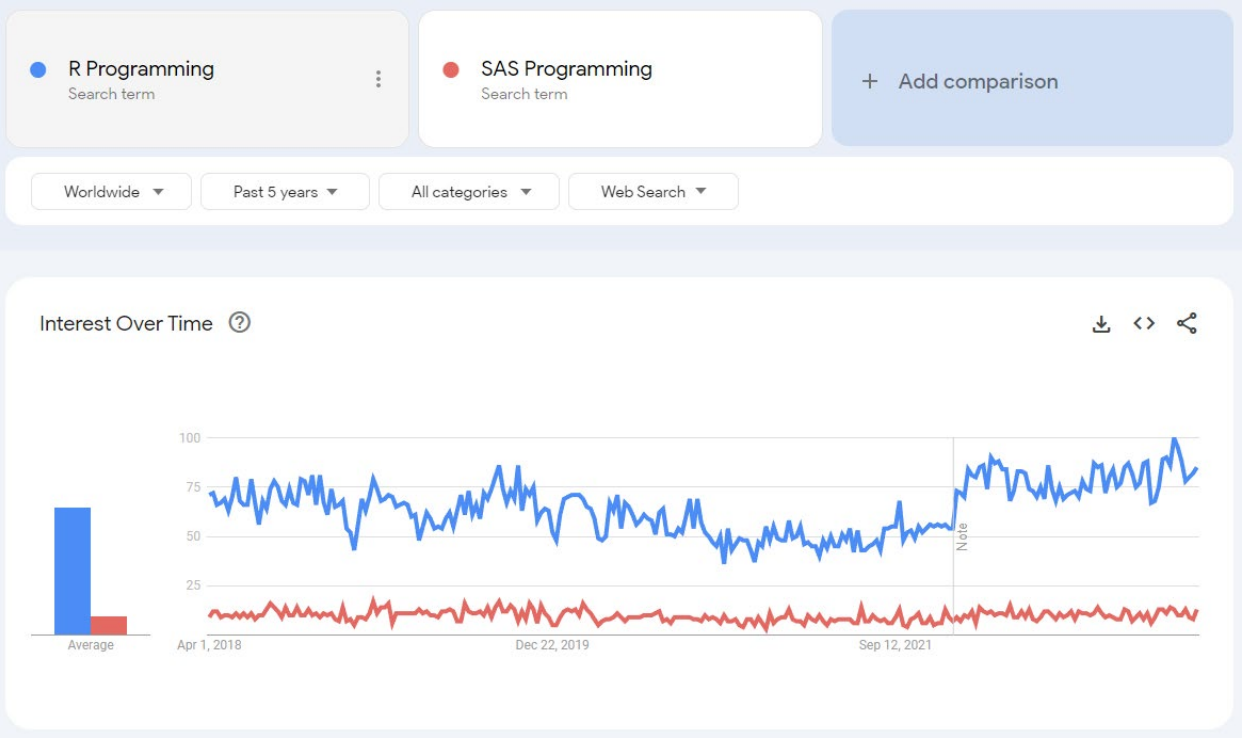

Figure 1. Google Trends SAS vs. R

DIFFERENCES BETWEEN SAS AND R

SAS and R are both widely used statistical software in the research and data science industry.

SAS is a commercial software that can be used to perform a variety of advanced analytics tasks, including data management, predictive analytics, and business intelligence. It can be accessed via a graphical interface as well as through SAS programming language.

On the other hand, R is an open-source language and environment for statistical computing and graphics. It offers a wide range of statistical techniques such as time-series analysis, linear and nonlinear modeling, classification, and clustering. The most popular IDE for R is RStudio, developed by RStudio PBC.

While SAS has been the go-to software for decades, the rising popularity of R among researchers and data scientists is a testament to its flexibility and the growing demand for open-source solutions. Each

software has its own unique advantages and choosing the right one for a specific task depends on the project requirements and the user's preferences.

MAIN DIFFERENCE SHEET

	SAS Programming	R Programming
Analytic Interface	SAS Studio	R Studio
Data Processing	Row by Row	Column by Column
Processing Structure	DATA / PROC steps	Single statement (Expressions with functions)
Function	Macros	function
Macro variables	Yes	No
Data Structure	SAS datasets	Data Frame
Object-oriented	No	Yes
Customer Support	Dedicated Support	Community Support
Feature Update	Less Frequent	Very frequent
Speed	Fast	Very fast
System	Closed System, thoroughly validated	Open System, transparent but validation could be questioned

PROGRAMMER SYNTAX

A QUICK EXPLANATION

The goal of this paper is to help SAS clinical programmers learn more about R and the R programming language. Learning the R programming language syntax is crucial and considered the first step for any programmer.

In this paper, the examples use `tidyverse`, which is a collection of R packages designed for data science. All packages within tidyverse follow the same design philosophy, grammar, and data structures. Compared to the function and usage concept of base R, tidyverse packages are simpler, clearer, and more logical for data processing and visualization. This makes it easier for beginners to start learning R by focusing on tidyverse directly, even if they decide to give up using base R.

To demonstrate common operations in both SAS and R, this paper provides examples that use `adsl`. By default, the following code operates in the SAS 'Work' library and in the R 'working directory'. The key syntax for implementing the functions is highlighted in **bold**.

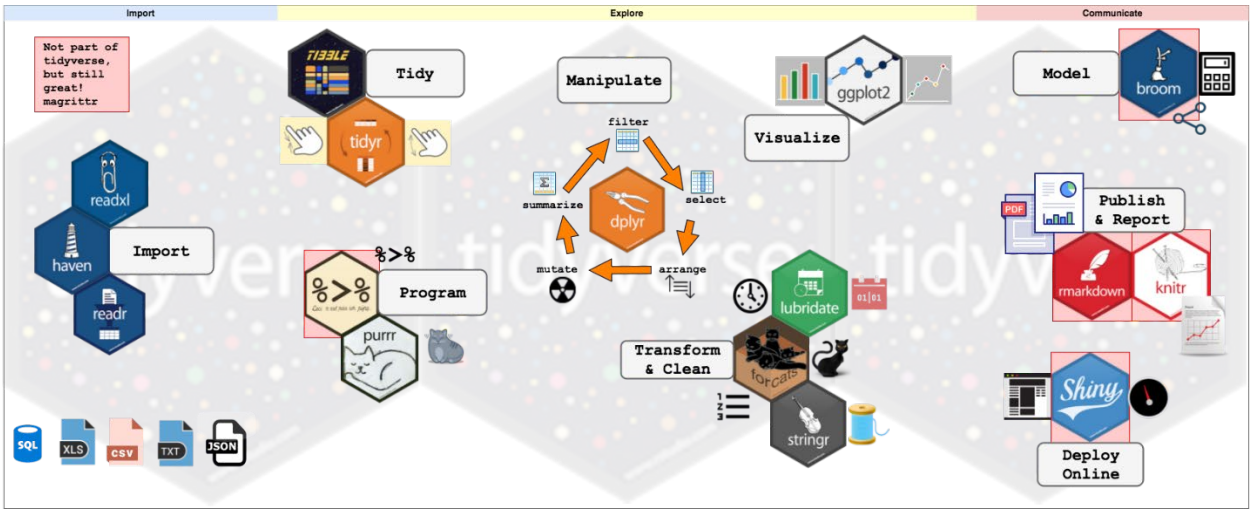

Figure 2. tidyverse

FILE OPERATIONS

Purpose	Syntax in SAS	Syntax in R
Define file locations	**libname** work "location"; data work.ds_1; **set** ds_raw; run;	**saveRDS**(ds_1,file="location/ds_raw.rds") *or* **setwd**("location") **saveRDS**(ds_1,file="ds_raw.rds") ds_1<- **readRDS**("location/ds_raw.rds") *or* **setwd**("location") ds_1<-**readRDS**("ds_raw.rds")
Export	**proc export** data=my_data outfile="my_file.csv" dbms=csv replace; run;	**write_csv**(my_data, "my_file.csv")
Import	**proc import** datafile="my_file.csv" **out=**my_data dbms=csv; run;	my_data <- **read_csv**("my_file.csv")

DATASETS (NEW, COMBINING & FILTERING)

Purpose	Syntax in SAS	Syntax in R
New dataset	**data** ds_new; **set** ds_old; run;	ds_new <- ds_old
Vertical combine	data ds_new; set ds_1 ds_2; run;	ds_new <- **bind_rows**(ds_1,ds_2)
Horizontal combine	data ds_new; **merge** ds_1(in=in_1) ds_2; **by** USUBJID; if in_1; run;	ds_new <- **left_join**(ds_1,ds_2, **by**="USUBJID")
Filter a specific value	data ds_new; set ds_old; **if** SEX**=**"M"; run;	ds_new <- ds_old %>% **filter**(SEX**==**"M")
Filter several values	data ds_new; set ds_old; **if** AGE **in** (80,90); run;	ds_new <- ds_old %>% **filter**(AGE **%in%** c(80,90))
Filter the first data for each group	data ds_new; set ds_old; **by** USUBJID; **if first.**USUBJID; run;	ds_new <- ds_old %>% **group_by**(USUBJID) %>% **slice(1)**
Filter out date-type data	data ds_new; set ds_old; **if** RANDDT>"01APR2023"d; run;	ds_new <- ds_old %>% **filter**(RANDDT>as.Date("2023-04-01"))

VARIABLES (DROP, KEEP, RENAME, NEW & CONDITIONAL EDITING)

Purpose	Syntax in SAS	Syntax in R

Drop keep	data ds_new(**keep**=USUBJID);
	  set ds_old(**drop**=SUBJID);
	run;
		ds_new <- ds_old %>%
		  select(-SUBJID) %>%
		  select(USUBJID)

Drop fixed prefix variables	data ds_new(**drop**=c:); set ds_old; run;	ds_new <- ds_old %>% **select**(**-starts_with**("c"))
Rename	data ds_new; set ds_old; **rename** old_name=new_name; run	ds_new <- ds_old %>% **rename**(new_name=old_name)
New	data ds_new; set ds_old; c=a+b; run;	ds_new <- ds_old %>% **mutate**(c=a+b)
Conditional editing	data ds_new; set ds_old; **if** RANDDT>0 **then** RANDFL="Y"; **else** RANDFL="N"; run;	ds_new <- ds_old %>% mutate(RANDFL= **if_else**(RANDDT>0,"Y","N"))
Conditional editing	data ds_new; set ds_old; **if** AGE<=18 **then** AGEGRP="0-18 years"; **else** if 18<AGE<=60 **then** AGEGRP="19-60 years"; **else** AGEGRP=">60 years"; run;	ds_new <- ds_old %>% mutate(AGEGRP= **case_when**(AGE<=18 ~"0-18 years", AGE<=60 ~"19-60 years", TRUE ~">60 years"))

STRINGS

Purpose	Code in SAS	Code in R
Keep string containing fixed values	data ds_new; set ds_old; if **find**(ARMCD,"ARM-A"); run;	ds_new <- ds_old %>% **filter**(**str_detect** (ARMCD,"ARM-A"))
Keep string containing a fixed prefix	data ds_new; set ds_old; if ARMCD**=:**"ARM-A"; run;	ds_new <- ds_old %>% **filter(str_detect** (ARMCD,"**^**ARM-A")) *<$ for end: ARM-A$>*
Keep positions 4-7	data ds_new; set ds_old; str=**substr**(USUBJID,4,4); run; *<start>, <length>*	ds_new <- ds_old %>% **mutate**(str= **str_sub**(USUBJID,4,7)) *<start>, <end>*
Replace the sub-strings in a string with other values	data ds_new; set ds_old; DTC1=**tranwrd**(DTC,"22","23"); run;	ds_new <- ds_old %>% **mutate**(DTC1=**str_replace_all**(DTC,"22","23"))
Connecting a few strings	data ds_new; set ds_old; USUBJID=**catx**("-", SITEID,SUBJID);	ds_new <- ds_old %>% **mutate**(USUBJID=**str_c**(SITEID,SUBJID,**sep**="-"))

	run;	
Keep the first word	data ds_new; set ds_old; reason=**scan**(other,1); run;	ds_new <- ds_old %>% mutate(reason=**word**(other,1))
Extract numbers	data ds_new; set ds_old; num=compress(other,,"dk"); run;	ds_new <- ds_old %>% mutate(num= str_extract(other,"\\d*")) *<String operations are very rich, no longer enumerated, please explore by yourself>*

SORTING

Purpose	Syntax in SAS	Syntax in R
Sorting in a certain order	proc sort data=ds_old out=ds_new; **by** USUBJID **descending** RANDDT; run;	ds_new <- ds_old %>% **arrange**(USUBJID, **desc**(RANDDT))
Remove repeat rows	proc sort data=ds_old **nodup**; **by** USUBJID RANDDT; run;	ds_old <- ds_old %>% **arrange**(USUBJID,RANDDT) %>% **distinct( )**
Reserve the first observation for each group	proc sort data=ds_old **nodupkey**; **by** USUBJID; run;	ds_old <- ds_old %>% **arrange**(USUBJID) %>% **group_by**(USUBJID) %>% **slice(1)**
Reserve the first observation of each group in a specified order	data ds_new; set ds_old; **by** USUBJID descending RANDDT; **if first.**USUBJID; run;	ds_new <- ds_old %>% **group_by**(USUBJID) %>% **slice**(**which.max**(RANDDT))
Receive data using Lag()	data ds_new; set ds_old; prev_ID=**lag**(USUBJID); run;	ds_new <- ds_old %>% mutate(prev_ID= **lag**(USUBJID,1))
Add a serial number to each row	data ds_new; set ds_old; **by** USUBJID; count**+1**; if **first.**USUBJID then count=1; run;	ds_new <- ds_old %>% group_by(USUBJID) %>% mutate(count=**row_number**( ))

CREATING YOUR OWN FUNCTIONS

Purpose	Syntax in SAS	Syntax in R

Making a new variable	**%macro** add_var(ds);
	  data &ds;
	  set &ds;
	  new_var=1;
	  run;
	%mend;
		add_var <- **function**(ds){
		  ds <- ds %>%
		  mutate(new_var=1)
		  return(ds)
		}
		my_data <- add_var(my_data)

	`%add_var(my_data);` *<Macros>*	*<Expressed in R functions>*

AFTER THIS, WHAT SHOULD WE LEARN

USING PUBLICLY AVAILABLE PACKAGES

In the field of clinical trial data analysis, there are a number of R packages that are available to the public and have been used and tested by many people. For example, the {tidyCDISC} package provides functions for exporting data in CDISC formats, while the {R4DSXML} package provides functions for importing CDISC Dataset-XML and Define-XML as R data frame. Additionally, the {clinicaltrialsummary} package can be used to generate summary tables and figures for clinical trial data, and the {ggplot2} package is a popular choice for creating high-quality graphics. Other useful packages for clinical trial data analysis include {tidyverse}, {plyr}, {reshape2}, and {data.table}. These packages can help streamline the analysis process and provide reliable and reproducible results.

HYBRID PROGRAMMING WITH SAS AND R

Hybrid programming with SAS and R has many benefits. It allows programmers to leverage the strengths of both languages and use them together seamlessly. For example, SAS is well-known for its powerful data manipulation capabilities, while R is widely recognized as a leading language for statistical analysis and visualization. By combining these two languages, programmers can take advantage of SAS's data manipulation and management capabilities while using R for data analysis and visualization. Hybrid programming solutions, such as {sasr}, {jupyter}/{quarto}, {saspy}, and {SASmarkdown} in R, make it easy to generate dynamic outputs from multiple languages. These solutions allow programmers to work with data in whichever language they are most comfortable with, and to switch between languages as needed to get the job done efficiently and effectively.

CREATING YOUR SPECIFIC PACKAGES

By making packages in R, businesses can make their own tools for analyzing and displaying data that meet their specific needs. These packages are collections of code that perform specific tasks and can be shared within the organization or even made publicly available on CRAN. Companies can create their own packages that integrate their internal data sources and workflows. This allows for greater efficiency and accuracy in data analysis and reporting. With the flexibility of R, the possibilities for creating custom packages are endless.

COLLECTING USE CASES FOR R – ESPECIALLY DATA DERIVATION AND VALIDATION

SAS programmers want to use R! There is lots of attendance on R training courses and lots of engagement in discussion around R. But there are truly few actual R outputs. I was thinking that most of us lack use cases for R. In the clinical trial data analysis field, there is a great need for more real-world examples of R programming applications. In particular, use cases for data derivation and validation are especially important.

By collecting more use cases and sharing them within the industry, we can help to bridge the gap between theoretical R knowledge and practical implementation. This will enable statistical programmers to become more familiar with R and to confidently use it in their work. Data derivation and validation are key areas where R can be particularly useful, and more case studies demonstrating this would be extremely beneficial. By showcasing successful examples of R in action, we can encourage greater adoption of this powerful tool in the clinical trial programming community.

R FOR SUBMISSION

On Nov 22nd, 2021, the R Consortium R Submissions Working Group successfully submitted an R-based test submission package through the FDA eCTD gateway. The submission package has been received by the FDA staff who were able to reproduce the numerical results. This submission was an example submission package following eCTD specifications which include a proprietary R package, R scripts for

analysis, R-based analysis data reviewer guide, and other required eCTD components. To our knowledge, this is the first publicly available R-based, or open-source-language-based FDA submission to the eCTD gateway for CDER. We hope this submission package and our learnings can serve as a good reference for future R-based regulatory submissions. The R-based pilot missions and review process went smoothly because the R Consortium group communicated regularly and in advance with the FDA regarding the commission. And with the recent successes by the R Consortium and enhanced collaboration with the FDA, R is trending toward higher standardization to satisfy regulatory needs.

CONCLUSION

While SAS still holds value for many users, the open-source packages available in R are quickly becoming the standard for the new workforce. With the recent successes of the R Consortium and better collaboration with regulatory agencies, R is moving in the direction of more standardization. This makes it an even better choice for analyzing and submitting data. Learning R is an investment in your future success, whether you want to improve your own skills or those of your team. Embrace the flexibility, efficiency, and simplicity of R programming and make sure you stay ahead of the curve. Don't let yourself be left behind in the dust!

REFERENCES

Pre-{admiral} Hackathon Workshop: Introduction to R for SAS programmers. 2023. https://www.cdisc.org/events/webinar/pre-admiral-hackathon-workshop-introduction-r-sas-programmers
R/Pharma. 2023. R/Pharma 2022 playlist. December 2022. https://www.youtube.com/c/RinPharma/
R/Pharma. 2023. R pilot submissions to FDA. November 2022. https://rinpharma.com/publication/rinpharma_268/
SAS vs R Programming: Which to Choose and How to Switch. Ivan Millanes. 2022. https://appsilon.com/sas-vs-r-programming/

ACKNOWLEDGMENTS

The authors would like to thank Sanofi for facilitating and supporting this paper.

RECOMMENDED READING

- *R Programming for Data Science - Roger D. Peng*
- *R for Data Science – Hadley Wickham & Garrett Grolemund*
- *R packages - Hadley Wickham & Jenny Bryan*
- *Advanced R - Hadley Wickham*
- *SAS and R - Bayer Oncology SBU https://bayer-group.github.io/sas2r/*
- *Introduction to R for SAS programmers - Sadchla Mascary, Stefan Thoma, Zelos Zhu, Thomas Neitmann https://pharmaverse.github.io/intro-to-r-for-sas-programmers-workshop/*

CONTACT INFORMATION

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

Sookie Kong
Sanofi
(+86) 17621227696
17621227696@126.com