PharmaSUG 2020 - Paper DV-299

Effective Exposure-Response Data Visualization and Report by Combining the Power of R and SAS Programming

Shuozhi Zuo, Hong Yan, Pharmacometrics, Regeneron Pharmaceutical, Tarrytown, NY, USA

ABSTRACT

Understanding the relationship between exposure and response is critical to finding a dose that optimally strikes a balance between drug efficacy and adverse events, therefore comprehensive exposure-response (E-R) analyses are highly demanded throughout all phases of clinical trials. This type of analysis could be planned, ad-hoc or exploratory, and it requires high quality data visualization and fast turnaround for dose selection, phase 2 decision or regulatory interaction and submissions etc. This paper introduces an innovative efficient way for exposure-response analysis results visual presentation by combining the power of SAS programming in data manipulation and processing and R language in data visualization and dynamic statistical analysis. We will use the logistics regression E-R analysis as an example, explaining how the data quality and accuracy is maintained in SAS and connecting R function to generate multiple figures by different exposure parameters and endpoints. This paper will describe the process in detail from input analysis data sets to final outputs including reading SAS data sets into RStudio, combining ggplot2 package with statistical analysis, reading titles and footnotes from excel sheet for each output dynamically, applying rtf package and visual basic scripting in RStudio for both RTF and PDF format, defining R function like SAS macro, and batch run all the R programs at once to update the multiple outputs.

INTRODUCTION

Understanding the Exposure-Response (E-R) relationship between Pharmacokinetics (PK) and Pharmacodynamics (PD) is key to the development and approval of each drug, PK and PD data contribute to much of what is on a drug package insert. Visualizing the data is a crucial part of exposure-response analyses to communicate the information clearly and efficiently. These analyses can be planned, but often arise as ad-hoc, exploratory or post-hoc right after clinical database lock. The volume of complex data visualization could be extensive to looking into many different exposure metrics verses various of endpoints by different methods and subgroups. The clinical team or management expect to communicate the analysis results in timely manner from Pharmacometrics department for decision making. Performing such tasks, the statistical programming team usually need to write or tweak the lengthy SAS programs which could be time-consuming, error-prone and tedious to modify graphic template for better presentation. The Pharmacometricians often use RStudio for exploratory data analysis and graphic presentation for internal purposes, however maintaining data traceability, results verification and formatting the outputs to be submission ready seems challenge.

This paper is to address the challenges forth mentioned by combing the power of SAS and R language after further exploring the existing R packages and functions. To facilitate the efficient figures validation and maintain the regulated programming process, a SAS macro is developed to create the analysis ready data sets prior to figures generation. Given that logistic and linear regression techniques are two of the most popular types of regression models utilized today, we used logistic regression of exposure-safety analysis as example to demonstrate our innovative approach.

OVERVIEW OF APPROACH

- Created ADaM exposure-response data sets in Basic Data Structure using SAS which contain observed drug concentration, Population PK model predicted exposure metrics and time to event elements of adverse events at interest etc.
- Developed a SAS macro generating analysis ready data sets by exposure metrics and response endpoints as input data sets in RStudio.
- Define a R function that read in analysis ready data sets, produce corresponding figures and perform the statistical analysis.

- Read in the titles and footnotes dynamically.
- Format and control outputs type.
- Batch run R codes for bulk of subset outputs.

Figure 1 summarize the overview of approach in a flow chart.

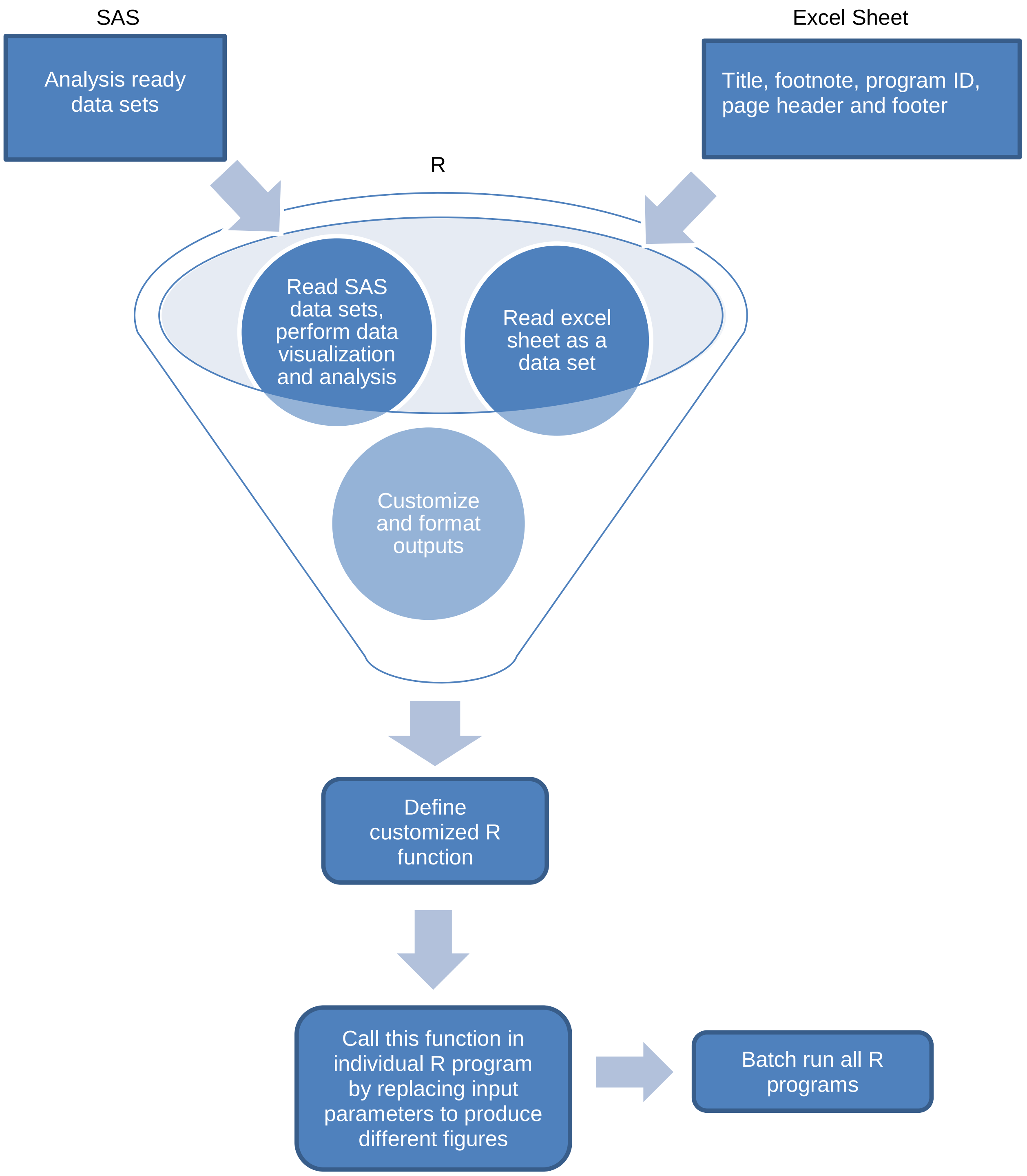

SAS
Analysis ready data sets
Excel Sheet
Title, footnote, program ID, page header and footer
R
Read SAS data sets, perform data visualization and analysis
Read excel sheet as a data set
Customize and format outputs
Define customized R function
Call this function in individual R program by replacing input parameters to produce different figures
Batch run all R programs

Figure 1. Flow Chart

PERMANENT SAS DATA SETS FOR FIGURE GENERATION

ADaM E-R analysis data sets (BDS) was programmed in SAS by combining the patient observed exposure, population PK model predicted exposure metrics and the time to events analysis. We then developed a SAS macro to create the quartiles, quartile range and mean of each quartiles of multiple exposure parameters over response endpoints by subgroups, mean probability of events and corresponding confidence intervals. These data sets are stored as permanent data sets for validation convenience and as input data sets in RStudio.

Sample Data 1 is an analysis ready data set for scatter plot and logistics regression analysis.

USUBJID	TRT01A	PARAMCDR	PARAMCD	AVAL	AVALR	TRT
XXXX-XXX-0001	300 mg Q4W	SAE	CTROUGH	141	0	300 mg Q4W (N = 100)
XXXX-XXX-0002	100/200 mg Q2W	SAE	CTROUGH	66.2	1	100/200 mg Q2W (N = 92)
…	…	…	…	…	…	…

Sample Data 1. Analysis Ready Data Set (1)

The trough concentrations and the probability of the adverse event are categorized by quartile of concentrations. AVAL is the mean trough concentrations of each quartile, MEAN is the mean probability of the adverse event at interest of each quartiles, UPPER and LOWER are the confidence interval for mean probability of the adverse event of each quartile.

Sample Data 2 is an analysis ready data set for bar plot.

QTR	N	MEAN	UPPER	LOWER	AVAL
Q1	46	0.196	0.315	0.077	34.078
Q2	49	0.143	0.244	0.041	58.000
Q3	48	0.125	0.222	0.028	82.521
Q4	49	0.061	0.131	-0.008	120.357

Sample Data 2. Analysis Ready Data Set (2)

R PROGRAMMING AND STATISTICAL ANALYSIS

INSTALLED R PACKAGES AND SETUP R WORKING ENVIRONMENT

R packages need to be installed only once but reloaded always at a new session using “library” function. The “haven” package reads SAS data sets into RStudio. The “ggplot2” package performs data visualization which breaks up graphs into semantic component such as scales and layers. The “readxl” package extracts the titles and footnotes from the excel sheet. The “rtf” package enables packing the figure from ggplot2 with the title and footnote into an RTF file, and “RDCOMClient” packages convert the RTF to a PDF format.

```
install.packages(c("haven","ggplot2","readxl","rtf"))
install.packages("RDCOMClient",repos="http://www.omegahat.net/R")
library(haven)
library(ggplot2)
library(readxl)
library(rtf)
library(RDCOMClient)
```

Sample Code 1: Installed and Loaded Packages

READ SAS DATA SETS INTO RSTUDIO

```
rdata<-read_sas(data)
rdata_stat<-read_sas(data_stat)
```

Sample Code 2: Read Analysis Ready SAS Data Sets into RStudio

PLOTTING AND STATISTICAL ANALYSIS

The ggplot2 package is a system for declaratively creating graphics, based on The Grammar of Graphics. After import data sets in RStudio, we need to tell ggplot2 how to map variables to aesthetics, what graphical primitives to use etc.[1] There are three main factors should be defined in ggplot2, a data frame, the aesthetics which define the x-axis, y-axis, colors, sizes and shapes of graphical element by variable in the data frame and geometrical objects – the actual graphical elements to display.[2]

In our analysis, the exposure was assigned to x-axis and the response was assigned to y-axis, further defined the shape and color by treatment group for the scatterplot. To increase the data visual effect, we jittered individuals experiencing adverse events and none adverse events are at the top and at the bottom in the figure respectively. You can also manually customize the color and shape for the scatterplot by treatment group. Just a few lines of R codes in Sample Code 3 give you a nice scatter plot already in Figure 2.

```
p<-ggplot()+
   geom_point(data=rdata,aes(x=AVAL,y=AVALR,shape=TRT,color=TRT),
   position=position_jitter(width=0.3,height=0.03),size=1.5,alpha=0.5)+
   scale_shape_manual(values=c(0,1))+
   scale_color_manual(values=c("red","green"))
```

Sample Code 3: Scatter Plot

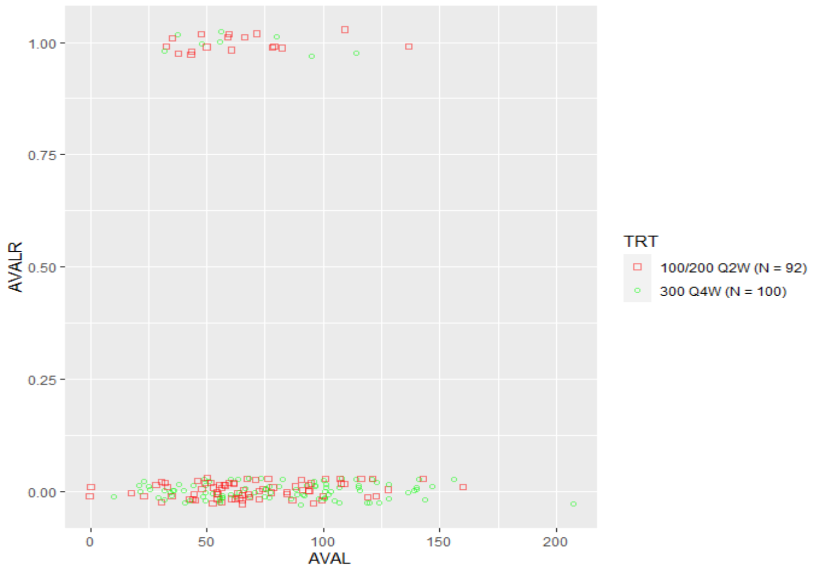


Figure 2. Scatter Plot

Now, we want to show the relationship or the trend between exposure and response, thus a logistic regression curve has been added to the Figure 2 by defining the binomial distribution (adverse event happens or not, 1/0) for general linear model with 95% confidence level in Sample Code 4. Regression line is colored in blue and standard error around regression line in grey in Figure 3.

```
p<-p+geom_smooth(data=rdata,aes(x=AVAL,y=AVALR),formula=y~x,
    method="glm",level=0.95,method.args=list(family="binomial"))
```

Sample Code 4: Logistic Regression

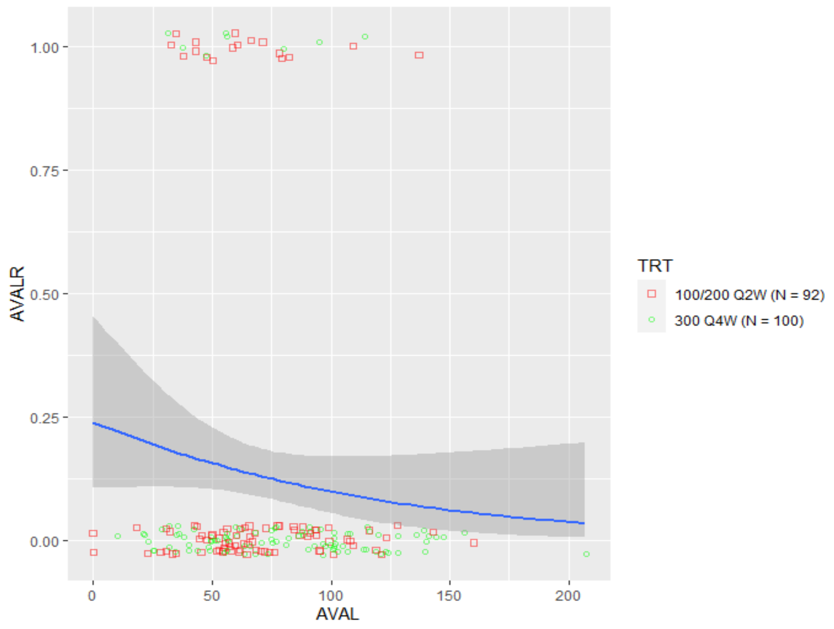


Figure 3. Logistic Regression

After seeing the logistic regression trend, the question arises on mean exposure by quartiles and mean probability of adverse event to further demonstrate the relationship. We add these data into Figure 3 in Sample Code 5. The confidence intervals are the green vertical lines overlap with the mean of response (circle), these vertical lines are placed at the means of interquartile ranges of exposure on the x-axis in Figure 4.

```
p<-p+geom_errorbar(data=rdata_stat,aes(x=AVAL,ymin=LOWER,ymax=UPPER),
    width=2,size=1,color="green")+
    geom_point(data=rdata_stat,aes(x=AVAL,y=MEAN),alpha=0.8,shape=21,
    size=1.5,fill="white")
```

Sample Code 5: Bar Plot for Quartile Analysis

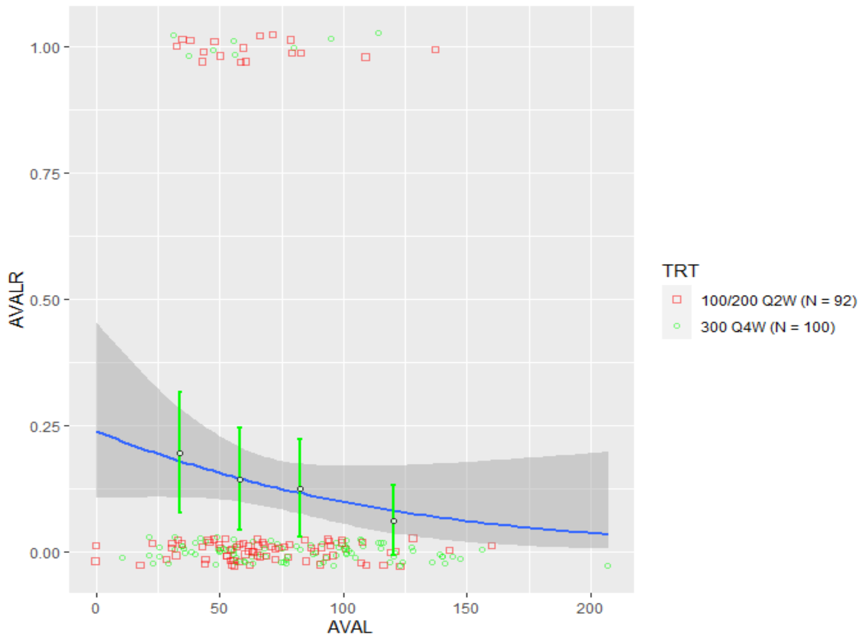


Figure 4. Bar Plot

The following R codes in Sample Code 6 are used to further polish the background color, scales, axis labels, legend position, font style and size for the best appearance in Figure 5.

```
p<-p+theme_bw()+theme(panel.border=element_rect(colour="black",size=0.8))+
   scale_x_continuous(limits=c(min,max),breaks=seq(min,max,int))+
   labs(x=xlab,y=ylab)+
   theme(legend.position="bottom",
   legend.direction="horizontal",legend.title=element_blank(),
   legend.text=element_text(size=7),legend.key.size=unit(1, "lines"))
```

Sample Code 6: Figure Appearance

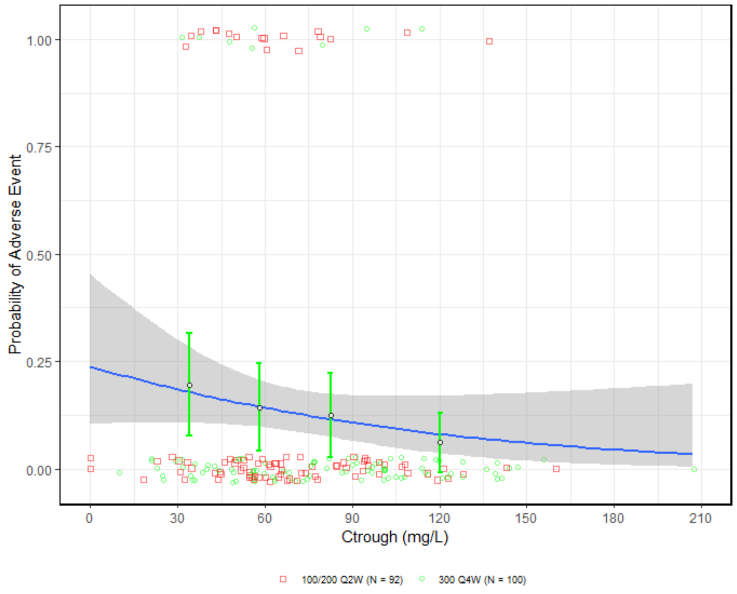


Figure 5. Final Exposure Response Analysis Figure

EXTRACT STATISTICAL COEFFICIENTS AND OUTPUT TO TITLE DYNAMICALLY

For statistical model results validation and reviewing convenience, we extract the key statistical results of each analysis and saved as the permanent data, output the coefficient (e.g.: P value) to a macro variable in the title and footnote as needed.

```
fit<-glm(rdata$AVALR~rdata$AVAL,family=binomial(link="logit"))
pval<-round(coef(summary(fit))[2,4],4)
```

Sample Code 7: Extract Statistical Coefficients

The “return” function can be used to show Output 1 in the log to check full statistical results for individual logistical regression analysis. Please remember to put the “return” function in the end of R program, otherwise R code will stop executing after the “return” function called.

```
return(summary(fit))
```

Sample Code 8: Return Statistical Results in the Log

```
Deviance Residuals:
    Min        1Q     Median         3Q        MAX
-0.7379   -0.5694    -0.5094    -0.4197     2.3143

Coefficients:
              Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)  -1.161986    0.499448   -2.327    0.020 *
rdata$AVAL   -0.010546    0.006854   -1.539    0.124
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null Deviance: 148.52 on 191 degrees of freedom
Residual Deviance: 145.99 on 190 degrees of freedom
AIC: 149.99

Number of Fisher Scoring iterations: 5
```

Output 1. Full Statistical Results

HANDLING THE REPORT TITLE AND FOOTNOTE DYNAMICALLY

It is always a good idea to maintain the titles and footnotes in a central location (i.e.: Sample Data 3) and convert it to a data frame for reporting purpose. We defined R code in Sample Code 9 to read the excel sheet containing output ID, titles and footnotes, then extract the title and footnote from the data frame into the corresponding outputs. Additional page header and footer information are documented in the file, read in and populated in the outputs following the company standard for the page header and footer and maintain the traceability.

PGM_NAME	HEADER	TITER1	FOOTNOTE1	FOOTER
F_PCLREG_AE	Regeneron Pharmaceutical, Inc. Protocol:REGN-XXX	Logistic Regression of Adverse Event with Trough Concentration at Week XX in Patients on REGN Treatment	Note: BLQs were set to 0.	/Data/Production/PK/Analysis/TFL/Figures/
F_AUC0TLREG_AE	Regeneron Pharmaceutical, Inc. Protocol:REGN-XXX	Logistic Regression of Adverse Event vs AUC0-t in Patients on REGN Treatment	AUC0-t are based on popPK prediction.	/Data/Production/PK/Analysis/TFL/Figures/

Sample Data 3. Excel Sheet in Central Location

```
titfoot<-read_excel("O:/Data/PK/TitleFootnote.xls")
titfootf<-titfoot[which(titfoot$PGM_NAME==fname),]
datetime<-Sys.time()
```

Sample Code 9: Extract Information from the Excel Sheet as a Data Frame in R

DEFINE FINAL OUTPUT FORMAT

RTF: Now all the elements including figure objects, statistical coefficients, titles and footnotes are taken cared. The rtf package is used to create the RTF or word file while combining the title, footnote and statistical coefficient with the Figure 5 from ggplot2. The format and layout can be adjusted and modified using the options and functions embedded in the package.

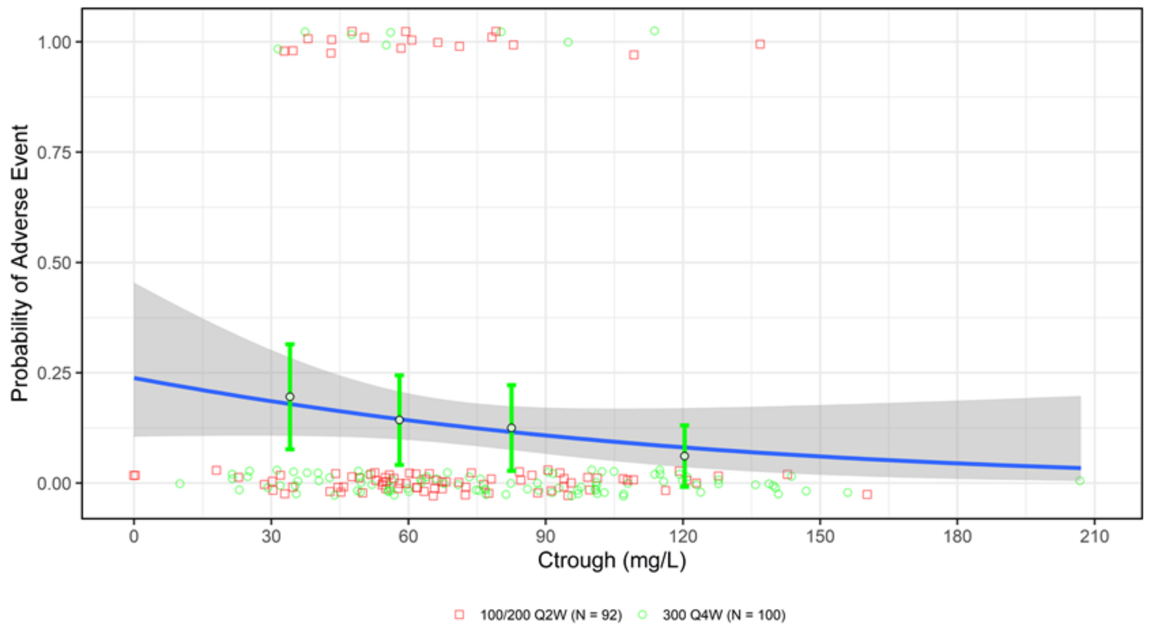
```
rtf<-RTF(paste0(fname,".rtf"),width=10,height=11,omi=c(1,1,1,1),
     font.size=9)
     addHeader(rtf,title="",subtitle=titfootf[1,2])
     startParagraph(rtf)
     #Here we concatenate the p-value with the title together
     addText(rtf,paste0(titfootf[1,3]," (PKSAF, P-value=",pval,")","\n"))
     endParagraph(rtf)
     addPlot(rtf,plot.fun=print,width=8,height=4.5,res=1000,p)
     addHeader(rtf,title="",subtitle=paste0(titfootf[1,4],"\n",
     titfootf[1,5],titfootf[1,1],".R (",user," ",datetime," R 3.5.1)."))
     done(rtf)
```

Sample Code 10: rtf Package

Output 2 is a screen capture of final output in RTF format.

Regeneron Pharmaceutical, Inc.
Protocol:REGN-XXX

Logistic Regression of Adverse Event with Trough Concentration at Week XX in Patients on REGN Treatment (PKSAF, P-value=0.1239)



Note: BLQs were set to 0. Regression line-blue, standard error around regression line-grey, jittered individuals experiencing no adverse events and adverse events are at the bottom and at the top of the figures, respectively-red and green dots. Means of response and confidence intervals (green vertical lines) around the means are presented in the figures by quantiles of exposure, these vertical lines are placed at the means of interquartile ranges of an exposure on the x-axis. /Data/Production/PK/Analysis/TFL/Figures/F_PCLREG_AE.R (shuozhi.zuo 2020-03-17 15:43:04 R 3.5.1).

Output 2. Final RTF Output

PDF: Clinical study report (CSR) writing requires all the outputs in both RTF and PDF format. Also, reviewers who are MacBook users may have some difficulty opening the RTF file, hence need PDF files. There are different ways to output figures in a PDF format in RStudio, but we found VBScript is the best option in terms of maintaining the content and layout of RTF file identical in a PDF file and taking no time. VBScript is known as a variant of MS visual basic language. One of the advantages of VBScript language is understandable for MS office applications. The use of VBScript in R programming can greatly enhance the capabilities to control office applications. Our idea is to utilize the dynamic interaction between RStudio and VBScript to generate the PDF file from RTF file. Sample Code 11 is the key script to execute the VBScript in R program.

```
file<-paste0("O:/Data/PK/",fname,".rtf")
wordApp<-COMCreate("word.Application")
wordApp[["Documents"]]$Open(Filename=file)
wordApp[["ActiveDocument"]]$SaveAs(paste0("O:/Data/PK/",
fname,".pdf"),FileFormat=17)
wordApp$Quit()
```

Sample Code 11: Execute the VBScript in R Program

DEFINE R FUNCTION TO PERFORM MUTIPLE SUBGROUP E-R ANALYSIS

Writing the similar program for each subgroup analysis can be tiresome and error-prone. To minimize the effort to write similar R codes repeatedly, we defined a customized R function that bear the similar concept to a SAS macro with the input parameters for repeated exposure-response logistics regression analysis by different exposure and endpoints. By calling this function in the individual R program, you can perform the analysis very easily and quickly, and you only need to maintain and global update this R function whenever you need to adjust and modify the figure appearance.

```
subsetting=function(data,data_stat,fname,xlab,ylab,min,max,int,user){
……
}
```

Sample Code 12: Define R Function for Efficiency

Call the R function in an individual program by entering the analysis data sets, name of the program, axis labels, scales and username.

```
source("O:/Data/PK/F_LREG.R")
subsetting(data="data.sas7bdat",data_stat="data_stat.sas7bdat",
fname="F_PCLREG_AE",xlab="Ctrough (mg/L)",ylab="Probability of Adverse
Event",min=0,max=210,int=30,user="shuozhi.zuo")
```

Sample Code 13: Individual Program for Subgroup Analysis

BATCH RUN OF R CODES

Opening and running individual R program one by one is very time-consuming and cumbersome when one need to run hundreds of E-R figures, so we explored the batch run option in RStudio and defined Sample Code 14 to read all R scripts in the same directory and run instantly which can save significant amount of time when performing dry run, production run, final run.

```
setwd("O:/Data/PK/")
directory_batch<-"O:/Data/PK/"
listrtf<-list.files(pattern="\\.R$")
print(listrtf)
for (k in 1:length(listrtf)){
    source(paste0(directory_batch,listrtf[k]))
}
```

Sample Code 14: Batch Run

Noticed: Sometimes, errors would come up from one of hundreds of outputs, which made the whole loop stop, and “tryCatch” function would tell RStudio to skip the error in the for-loop and keep generating the rest of the outputs. This function is very useful when you run hundreds of analysis outputs since sometimes it takes long time to complete, and you do not want the loop to stop because there are minor issues in one of hundreds of outputs. After all the R code have been executed, it will print out the error detail in the log so that you can check and update it afterward for individual outputs.

```
for(k in 1:length(listrtf)){
     tryCatch(source(paste0(directory_batch,listrtf[k])),
     error=function(e){cat("ERROR :",conditionMessage(e),"\n")})
}
```

Sample Code 15: Skip Error in the Loop When Batch Run

CONNECT R IN SAS

All the R codes in the example given are run in the RStudio, it is worth to mention that R functions can be executed in SAS program. A simply SUBMIT/ENDSUBMIT statement provides an interface to submit R codes to R statistical programming language within the PROC IML in SAS. To connect to R, specify the “R” option in the SUBMIT statement. The Sample Code 16 is straightforward in open code, but not so in SAS macro because the SUBMIT/ENDSUBMIT statements are not allowed in SAS macro.[7]

```
proc iml;
  submit/R;
  <R code>
  endsubmit;
quit;
```

Sample Code 16: Submit R Codes in SAS

PROC IML can also exchange data sets dynamically from SAS data set to R data frame or from R data frame to SAS data set to designated destination

```
proc iml;
  run ExportDataSetToR("WORK.SASdatasetName","RdatasetName");
quit;
```

Sample Code 17: SAS to R

```
proc iml;
  run ImportDataSetFromR("WORK.SASdatasetName","RdatasetName");
quit;
```

Sample Code 18: R to SAS

CONCLUSION

This paper presents a new approach which enables a SAS programmer with basic knowledge in R programming language to generate the high-quality exposure-response analysis figures without lengthy SAS program/macros, time-consuming in defining cosmetic SAS graphic template for better resolution, at same time maintaining consistency across all E-R analysis outputs and facilitating the validation of the data sets before plotting in RStudio. This new technique ensures the technical accuracy and quality. We hope that it can make your life a little easier when you work on bulk of exposure-response analysis figures with tight timelines. Additionally, you can apply this process flow of combining SAS and R programming for your own data presentation and analysis.

REFERENCES

[1] https://ggplot2.tidyverse.org/

[2] http://monashbioinformaticsplatform.github.io/2015-11-30-intro-r/ggplot.html

[3] Duncan Temple Lang. “The RDCOMClient package”. March 2012.
http://www.omegahat.net/RDCOMClient/

[4] Michael E. Schaffer. “Package ‘rtf’”. May 2019. https://cran.r-project.org/web/packages/rtf/rtf.pdf

[5] Hadley Wickham. “A Layered Grammar of Graphics”. Journal of Computational and Graphical Statistics. Volume 19, 2010. http://vita.had.co.nz/papers/layered-grammar.pdf

[6] Babych Oleksandr. “The Power of Data Visualization in R”, PharmaSUG, May 2019.

[7] Huei-Ling Chen, Zhen Zeng. “Application of R Functions in SAS to Estimate Dose Limiting Toxicity Rates for Early Oncology Dose Finding”, PharmaSUG, May 2019.

CONTACT INFORMATION

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

Hong Yan
Director of statistical programming, Drug Safety & Pharmacometrics, Regeneron Pharmaceutical
+1-914-847-1452
hong.yan@regeneron.com

Shuozhi Zuo
Senior statistical programmer, Drug Safety & Pharmacometrics, Regeneron Pharmaceutical
+1-914-418-0327
shuozhi.zuo@regeneron.com