PharmaSUG China 2024 - Paper CC-10082

Exploring the Interaction Between SAS and R to Enhance Programming Efficiency

Qi Zhai, Yu Li, Yi Yang, HENGRUI Co.

ABSTRACT

The integration of SAS and R has become increasingly prominent in data analysis and statistical modeling, offering the potential to enhance programming efficiency.

This paper investigates the symbiotic relationship between SAS and R, aiming to exploit their complementary strengths for improved analytical workflows. While SAS excels in robust data management, R provides a rich set of statistical tools and visualization capabilities. By leveraging their interoperability, researchers can seamlessly preprocess data in SAS and perform advanced analysis in R. The integration facilitates code and results sharing, fostering collaboration and reproducibility. Various integration strategies are explored, including calling R functions from SAS and transferring data between environments. Through practical examples, we demonstrate how this hybrid approach accelerates analysis.

Overall, integrating SAS and R offers a promising avenue for optimizing programming efficiency.

INTRODUCTION

SAS serves as a prevalent tool among biostatisticians and programmers in the pharmaceutical and biotechnology sectors for data processing and generating TFLs (Tables, Figures, and Listings). Those proficient in SAS can utilize Proc Template to create graphs and implement loops for specifying parameters, different groups, or individual plots. However, mastering the syntax for complex diagrams can pose challenges.

On the other hand, R ggplot2 can create pictures flexibly, and build complex plots in groups. R can finely adjust the details of the picture, and the operability is not as difficult as expected. By leveraging SAS's robust data processing capabilities alongside R's visualization tools, a wide array of visually appealing figures can be crafted. Even individuals unfamiliar with R can quickly grasp its functionality.

This paper will introduce the process of creating figures in batches with SAS and R, as well as tips in the integration of SAS and R, such as how to submit R code in SAS and how to implement loops.

WORKFLOW

Both SAS® Graph Template Language (GTL) and R ggplot2 are robust tools for data visualization. As a SAS programmer, you may encounter specific requirements such as achieving precise graph aesthetics or utilizing R for graph creation.

Therefore, instead of investing significant time and effort in learning R for these specific needs, SAS users can leverage existing tools to process data, generate necessary datasets, and subsequently invoke R to generate figures.

In our workflow, data processing occurs within SAS. Using SAS, all the data needed in the figure is obtained, and macro variables or macro programs are established. Submit R code to creates figures in R. Figures generated by R is ultimately stored in SAS as a .rtf file to produce the final output.

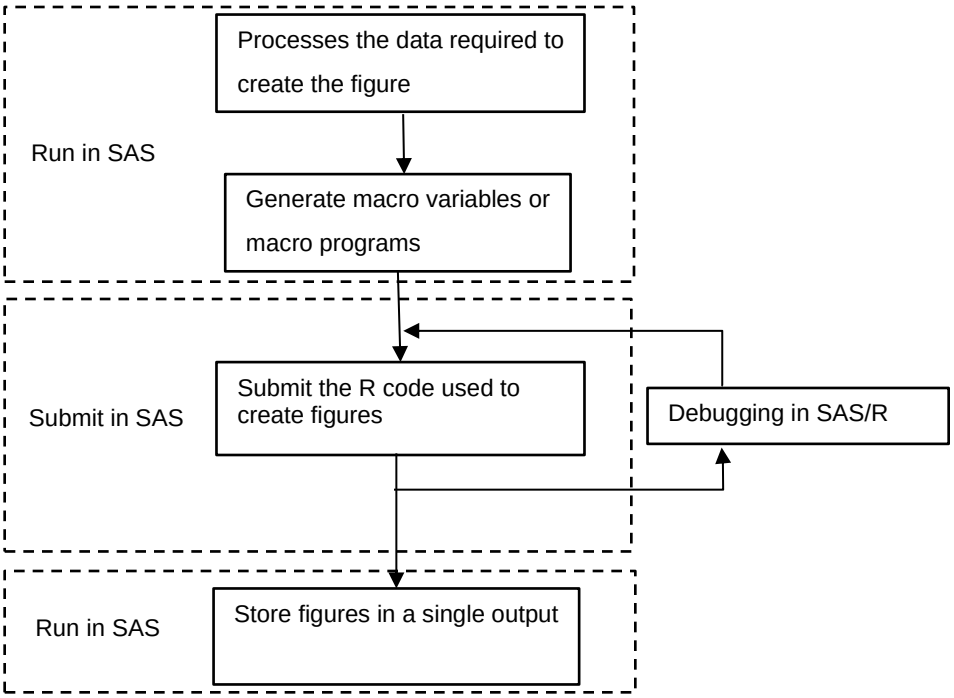

Figure1. Workflow.

THE WAY OF RUNNING R FROM SAS

There are several primary methods for integrating R with SAS. With official SAS support, you can submit R code in SAS via SAS/IML, or translate the PMML file generated by R model export into SAS code through SAS Model Manager. Due to the high requirements for the version of SAS, for users without access to the required SAS version, some users have independently developed and submitted R code in SAS through macro Proc R, or translated neural networks and decision tree models generated by R into SAS code.

Starting in SAS ® 9.3, the R interface enables SAS users on Windows and Linux who license SAS/IML ® software to call R functions and transfer data between SAS and R from within SAS. Potential users include SAS/IML users and other SAS users who can use PROC IML just as a wrapper to transfer data between SAS and R and call R functions.

First, SAS 9.3 or later must be running on a Linux or Windows operating system. SAS/IML and the corresponding version of R software must also be installed. For example, the 9.4M8 version of SAS corresponds to the download of the 4.0.0-4.4.0 version of the R software.

```
proc setinit; run;   /*Submit the SAS statement to determine if SAS/IML is installed*/
tar -xzvf R-4.0.0.tar.gz
cd R-4.0.0           /*Go to the CRAN image of R and unzip the downloaded source package*/
./configure --prefix=/usr/local/R/4.0.0 --enable-R-shlib
Make                 /*Run the command to configure and compile R*/
```

Second, download the R package. The download sources include: R official website (CRAN), Bioconductor, GitHub, and manual installation. Manual installation allows you to download the package from the Internet and import it to the path where the R package is stored (e.g. C:/Users/xxxx/Documents /R/win-library/4.1).

Third, Configure the SAS interface of R. The RLANG system option, which determines if you can call R from SAS, must be set to RLANG, not NORLANG. To determine if RLANG is enabled, submit the following SAS statement.

```
proc options option=rlang; run;                    /*Submit the SAS statement to determine
                                                     if RLANG is enabled */
RLANG Support access to R language interfaces      /*This log means that R commands can
                                                     be executed from SAS.*/
NORLANG Do not support access to R language interfaces    /*This log requires the
                                                     RLANG option to be added to the SAS
                                                     configuration file.*/
```

RLANG can only be set at invocation, not during your SAS session. The easiest way to do this is to include it in your SAS configuration file. To determine the location of your SAS configuration file, submit the following SAS statement.

```
proc options option=config; run; /*Submit the SAS statement to determine the location of your SAS
                                   configuration file */
```

Add the following setting to your SAS configuration file if it is not already present. Updating this file might require administrator privileges.

```
-rlang
```

```
eg:
```

```
-RLANG
-SET R_HOME "/usr/local/R/3.6.3/lib64/R"
/* the -RLANG had to be above the config reference for this to be recognized properly. */
```

Finally, restart SAS and check if SAS now supports access to the R interface.

So far, the computer used is already installed with SAS ,R and R packages, and supports access to the R language interface. We integrate R in SAS with the help of the process PROC IML and its functions via the SAS/IML interface. R code submitted through SAS essentially runs in R, so it fully supports various models and functions of R, and easy to understand in terms of data exchange and error resolution.

The following is an example of submitting R code in SAS to call the R package.

```
proc iml;
   call ExportDataSetToR("test");        /* this command sends test data set to R */
    submit / r;                          /* start executing R code */
   library(tidyverse)                    /* the first R statement  */
    library(svglite)                     /* the last R statement */
endsubmit;                               /* stop executing R code */
quit;                                    /* executing the QUIT statement would terminate the
                                            IML and R sessions and close the plot window. */
```

IMPLEMENT CIRCULARITY IN THE INTEGRATION OF SAS AND R

By defining macro variables in SAS, a cycle of specifying parameters for the same graph can be achieved, thus creating a series of graphs.

In the following example, we show how to create a series of figures using the familiar SAS to create a loop. We want individual data plots for six subjects per group per parameter.

In the first step, prepare the dataset in SAS, create a dataset with one row per subject, and loop it with parameters and groups. The dataset contains all the information needed to create a picture of the individual subjects in each group for each parameter, such as time point (x-axis), analysis value (y-axis), label, screening information (stage, group, parameter, etc.), as well as maximum and minimum y-axis values.

USUBJID	TRTA	TRTAN	PARAM	PARAMN	AVAL	STAGE	TPT	TPTN	IND	IND_H
001	A	1	T	1	0.001	S	V16	16	0.001	0.005
002	A	1	T	1	0.001	S	V15	15	0.001	0.005
003	A	1	T	1	0.001	S	V20	26	0.001	0.005
004	A	1	T	1	0.001	S	V19	19	0.001	0.005
005	A	1	T	1	0.005	S	V16	16	0.005	0.005
006	A	1	T	1	0.001	S	V15	15	0.001	0.005
001	A	1	S	2	0.004	S	V14	14	0.004	0.005
002	A	1	S	2	0.001	S	V11	11	0.001	0.005

Table 2. The dataset ind_final.

The second step is to create a dataset for looping. According to the parameters and grouping loops, there are two groups and 4 parameters, which need to be looped 8 times to finally get 8 figures.

```
proc freq data=ind_final noprint;
  tables trtan*paramn*param / out=param_01;    /*Loop with parameters and groups. There are 8
                                                combinations in total, and 8 figures should be created*/
run;

data  param_final;
  length txt $ 1000;
  set param_01;
  row = left(put(_n_, z2.));
  txt = cats('%nrstr(%f_ind(row=', row, '))');/*Generate a command to call the macro %f_ind.
                                                According to the combination of groups and
                                                parameters, a total of 8 lines, the macro program
                                                f_ind is called 8 times in the loop */
run;
```

TRTAN	PARAMN	PARAM
1	1	T
1	2	S
2	1	T
2	2	S
3	1	T
3	2	S
4	1	T
4	2	S

Table 2. The dataset param_01 shows the variables and combinations that need to be looped.

TXT	TRTAN	PARAMN	PARAM
%nrstr(%f_ind(row=01))	1	1	T
%nrstr(%f_ind(row=02))	1	2	S
%nrstr(%f_ind(row=03))	2	1	T
%nrstr(%f_ind(row=04))	2	2	S
%nrstr(%f_ind(row=05))	3	1	T
%nrstr(%f_ind(row=06))	3	2	S
%nrstr(%f_ind(row=07))	4	1	T
%nrstr(%f_ind(row=08))	4	2	S

Table 3. The TXT column in dataset param_final shows the commands that generate macros for loops

The third step is to create a macro for looping：f_ind.

```
%macro f_ind(row=);

data  _null_;
  set param_final;
  where (row eq "&row");
  call symputx("trtan", trtan);
  call symputx("param", param);
run;

data  ind_final_01;
  set ind_final;
  where (param eq "&param" and trtan eq &trtan);   /*A subset of data is determined based
                                                     on all filter criteria */
  ……                                              /*If the axis scale or label needs to be
  ……                                                 adjusted, adjust it accordingly*/
run;

proc iml;
  call ExportDataSetToR("ind_final_01");  /* this command sends ind_final_01 data set to R */
  r_path   = "xxx";                       /*Specify a path if needed*/
  r_dsnout = "xxx";                       /*Specify the output if desired*/
  %include "%sysfunc(pathname(xxxx))/fadpc_ind_r.sas";
/* R code is stored separately in this file and submitted via %include. */

ERROR: Submit block cannot be directly placed in a macro. Instead, place the
submit block into a file first and then use %include to include the file
within a macro definition.
/* To call R code in the SAS macro program, it must be submitted via %include. Otherwise, it will be reported to this
erroe*/
```

```
quit;                                   /* executing the QUIT statement would terminate the IML
                                           and R sessions and close the plot window. */
%mend  f_ind;


submit r_path r_dsnout / r;             /*The following code is the R code in the fadpc_ind_r.sas file*/
/*r_path is the value of a macro variable obtained through SAS's r_path  = "xxx" */
/*r_ dsnout is the value of a macro variable obtained through SAS's r_dsnout = "xxx". There is no concept of macro
variables in R like in SAS */
    library(tidyverse)
    library(ggsci)
    library(svglite)

p <- ggplot(data= ind_final_01,         /*R code for creating figure*/
    ......
    ......


    file_png <- "&r_path/&r_dsnout.png"    /*Output 8 PNG imagse to the specified path*/
    file_svg <- "&r_path/&r_dsnout.svg"    /*Output 8 the SVG images to the specified path*/

   ggsave(plot=p, file=file_png, dpi=300, height=15, width=22,
   units="cm")                          /*Set the output image size*/
    ggsave(plot=p, file=file_svg, dpi=600, height=15, width=22, units="cm")


  endsubmit;                            /* stop executing R code */
```

Also, view the dataset generated by R in SAS.

```
proc iml;
submit / R;
z = c('a','b','c','d','e')
RData <- data.frame(x=1:5, y=(1:5)^2, z=z)
endsubmit;

call ImportDataSetFromR("Work.MyData", "RData");
/* In SAS, importing a dataset from R. Ensure that a valid R script or command has been executed to generate the
data file before execution.*/

quit;
```

Finally, put these 8 figures in one .rtf in SAS.

```
data _png_99;
  length png $ 500;
  do i=1 to &png_n;                                  /* &png_n is resolved to 8 */
```

```
    row = put(i, z2.);
    png = cats("&esc", "S={height=15cm width=22cm just=center preimage=", '"',
pathname("xxx"), "/&dsnout._", row, ".png", '"', "}");/* In the variable png, the size, storage
    output;                                          path, and file name of the 8 figuress are specified */


  end;
run;

proc report data=_png_99 nowindows style=[rules=none frame=void cellpadding=0];
  column png;
  define png / display ' ' style={bordercolor=#ffffff};
run;                                  /* Display 8 images in the same RTF via the PROC REPORT */
```

The example above shows how to implement a loop in SAS to create an image with multiple calls to R code. For those who are familiar with R, it is also possible to use the usable loop structures or functions in R.

CONCLUSION

The integration of SAS and R expands users' options for solving data analysis and statistical modeling challenges. For those less familiar with R but interested in leveraging its powerful data visualization capabilities, initiating R code from within SAS is a viable starting point. After solving the problems of how to call R code and implement loops, users can flexibly process data and create figures. Spending less time learning R may bring greater convenience to people's subsequent work and adjust the existing work model.

Overall, integrating SAS and R is a significant way to improve the efficiency of optimization programming.

REFERENCES

Bruce Gilsen, Federal Reserve Board, Washington, DC. "Using the R interface in SAS ® to Call R Functions and Transfer Data", support.sas,Paper 3556-2019.

Diana Bulaienko, Experis Clinical, Kharkiv, Ukraine." SAS® and R - stop choosing, start combining and get benefits!", PharmaSUG 2016 - Paper QT14.

David Edwards, Bella Feng, Brian Schultheiss, Amgen, Inc."SAS and R Playing Nice Together". PharmaSUG 2017 - Paper PO22

R integration with SAS, URL:https://cn.bing.com/translator?ref=TThis&from=zh-Hans&to=en&isTTRefreshQuery=1

R language loop (for, while, repeat), URL:https://geek-docs.com/r-language/r-tutorials/g_loops-in-r-for-while-repeat.html

What versions of R are supported by SAS? URL:https://blogs.sas.com/content/iml/2013/09/16/what-versions-of-r-are-supported-by-sas.html

How do I download the R package? URL:https://gitcode.csdn.net/65aa2bb4b8e5f01e1e44bcc9.html

CONTACT INFORMATION

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

Name: Qi Zhai
Enterprise (optional): HENGRUI Co.
E-mail: qi.zhai.qz3@hengrui.com