SASPaper SAS104-2022

SAS® and Open Source Playing Nicely Together

Jim Box and Samiul Haque, SAS Institute Inc.

ABSTRACT

Open-source languages like R and Python are immensely popular and quite useful. Did you know you could write code blocks of Python and R inside of SAS programs? You can also invoke SAS® analytics from open-source programs. In this presentation, we will summarize all the ways SAS and open source

INTRODUCTION

SAS has had a long team relationship with open-source programs. PROC IML (Interactive Matrix Language) has enabled programmers to run R code in SAS for years. The new cloud-based SAS environment, SAS Viya, has enabled more opportunities to incorporate R program. It has also opened the door for several ways to integrate SAS with Python. This paper will explore some of the methods SAS programmers can use to get the most out of the intersection of SAS and open-source coding methods.

All SAS code shown in the paper was run in SAS Studio.

R AND SAS

R WITH SAS 9.4

The primary method of using R and SAS together has always been PROC IML. The basic code structure for using R in this manner is to call PROC IML, move SAS data into R, operate R code, then mode data back to SAS, if necessary:

```
PROC IML;
call ExportDataSetToR(<"SAS libname.dataset">,"<R data frame>");
submit / R;


R code block


end submit;
call ImportDataSetFromR("<SAS dataset to write>","<R data frame>");
quit;
```

A basic example of this would be:

```
1  PROC IML;
2  call ExportDataSetToR("Sashelp.Heart", "df" );
3
4  submit / R;
5     summary(df)
6  endsubmit;
7
8  QUIT;
9
```

Display 1. PROC IML Code

Note that in the R code block, the syntax is complete R code, even though the syntax highlighting might be using SAS keywords. Also note that there is no need for semi-colons, just use whatever R code is appropriate.

The output of this code will show up in the Results tab, where it will look just like R output.

Code Log Results

```
  Status                          DeathCause     AgeCHDdiag       Sex
Alive:3218   Cancer                    : 539   Min.   :32.0   Female:2873
Dead :1991   Cerebral Vascular Disease: 378   1st Qu.:57.0   Male  :2336
             Coronary Heart Disease    : 605   Median :63.0
             Other                     : 357   Mean   :63.3
             Unknown                   : 112   3rd Qu.:70.0
             NA's                      :3218   Max.   :90.0
                                               NA's   :3760
  AgeAtStart        Height          Weight        Diastolic
Min.   :28.00   Min.   :51.50   Min.   : 67.0   Min.   : 50.00
1st Qu.:37.00   1st Qu.:62.25   1st Qu.:132.0   1st Qu.: 76.00
Median :43.00   Median :64.50   Median :150.0   Median : 84.00
Mean   :44.07   Mean   :64.81   Mean   :153.1   Mean   : 85.36
3rd Qu.:51.00   3rd Qu.:67.50   3rd Qu.:172.0   3rd Qu.: 92.00
Max.   :62.00   Max.   :76.50   Max.   :300.0   Max.   :160.00
                NA's   :6       NA's   :6
   Systolic          MRW          Smoking         AgeAtDeath     Cholesterol
Min.   : 82.0   Min.   : 67   Min.   : 0.000   Min.   :36.00   Min.   : 96.0
1st Qu.:120.0   1st Qu.:106   1st Qu.: 0.000   1st Qu.:63.00   1st Qu.:196.0
Median :132.0   Median :118   Median : 1.000   Median :71.00   Median :223.0
Mean   :136.9   Mean   :120   Mean   : 9.367   Mean   :70.54   Mean   :227.4
3rd Qu.:148.0   3rd Qu.:131   3rd Qu.:20.000   3rd Qu.:79.00   3rd Qu.:255.0
Max.   :300.0   Max.   :268   Max.   :60.000   Max.   :93.00   Max.   :568.0
                NA's   :6     NA's   :36       NA's   :3218    NA's   :152
    Chol_Status     BP_Status        Weight_Status            Smoking_Status
Borderline:1861   High   :2267   Normal     :1472   Heavy (16-25)    :1046
Desirable :1405   Normal :2143   Overweight :3550   Light (1-5)      : 579
High      :1791   Optimal: 799   Underweight: 181   Moderate (6-15)  : 576
NA's      : 152                  NA's       :   6   Non-smoker       :2501
                                                    Very Heavy (> 25): 471
                                                    NA's             :  36
```

Display 2. PROC IML Results

Programs can include library statements to add packages to the R session, but the system administrator must be the one to install packages into the environment.

When creating R graphics, currently there is not a way for the graphs to show up in the output windows. To get the output, you'll have to write it out to a file location (e.g. , use the png function in *ggplot2* library).

There are also some R libraries that will allow you to read SAS datasets into R dataframes in the R environment.

Package	Details
sas7bdat	df=read.sas7bdat("heart.sas7bdat") No function for writing files
haven	Part of the tidyverse, includes functions for reading and writing SAS, SPSS and Stata files. • Read_sas() • Write_sas()

Table 1. R packages for interacting with sas7bdat datasets

```
> library(haven)
> df2 = read_sas("heart.sas7bdat")
> head(df2)
# A tibble: 6 x 17
  Status DeathCause AgeCHDdiag Sex   AgeAtStart Height Weight Diastolic Systolic   MRW Smoking
  <chr>  <chr>           <dbl> <chr>      <dbl>  <dbl>  <dbl>     <dbl>    <dbl> <dbl>   <dbl>
1 Dead   "Other"            NA Fema…         29   62.5    140        78      124   121       0
2 Dead   "Cancer"           NA Fema…         41   59.8    194        92      144   183       0
3 Alive  ""                 NA Fema…         57   62.2    132        90      170   114      10
4 Alive  ""                 NA Fema…         39   65.8    158        80      128   123       0
5 Alive  ""                 NA Male          42   66      156        76      110   116      20
6 Alive  ""                 NA Fema…         58   61.8    131        92      176   117       0
# … with 6 more variables: AgeAtDeath <dbl>, Cholesterol <dbl>, Chol_Status <chr>,
#   BP_Status <chr>, Weight_Status <chr>, Smoking_Status <chr>
```

Display 3. Using haven library to read SAS datasets into R

R AND SAS VIYA

All of the PROC IML code will run exactly the same in the Viya environment, so everything above works just the same. Additionally, there is an R package called SWAT (Scripting Wrapper for Analytic Transfer) that enables you to interface with the SAS Cloud Analytical Services (CAS), which are the in-memory data libraries that is at the center of the Viya platform. The SWAT package allows you to write an R program that will connect to a CAS server and analyze large in-memory datasets, then use the results in the R environment.

```
library(swat)

Sys.setenv(CAS_CLIENT_SSL_CA_LIST='/etc/pki/tls/certs/ca.crt')

username <- rstudioapi::askForPassword("username")
password <- rstudioapi::askForPassword("password")

session <- swat::CAS('https://viya4.globalhls.sashq-d.openstack.sas.com/cas-shared-default-http', 443, protocol='https',

currentCASlib<-'PUBLIC'

#list available tables in caslib
sites <- defCasTable(session,caslib=currentCASlib,"StudySites")
head(sites)
dim(sites)
```

```
> head(sites)
  Replication                          Site Start_Flag StartUp Cost StartDelay CountryDelay FirstPatient LastPatient
1           1     Chenango Memorial Hospital          1   22886 5500         60           32        22891       23298
2           1           Derry Medical Center          1   22890 4500         64           36        22894       23300
3           1        Lassen General Hospital          1   22891 6000         65           37        22895       23295
4           1          Sacred Heart Hospital          1   22894 5000         68           40        22898       23298
5           1 Shasta Regional Medical Center          1   22879 5500         53           25        22883       23297
6           1             St Eligus Hospital          1   22897 7500         71           43        22902       23295
  N_Screened N_Enroll     P_Fail N_Complete    P_Comp StudyVisits ScreenCost VisitCost TotalCost
1         93       84 0.09677419         68 0.8095238         236      62775    206516    269291
2        110       97 0.11818182         86 0.8865979         277      82500    275200    357700
3         93       80 0.13978495         64 0.8000000         223      60450    187200    247650
4        118      108 0.08474576         90 0.8333333         309      85550    293580    379130
5        100       81 0.19000000         64 0.7901235         230      70000    201600    271600
6         81       69 0.14814815         51 0.7391304         194      53460    151470    204930
```

Display 4. Connecting to CAS data libraries in R Studio via the swat library

```
site51 <- cas.dataStep.runCode(session,
                               code="
        data Site51;
        set PUBLIC.StudySites;
        if Replication = ' 51';
run;"
)
results<-cas.table.fetch(session,
                         table=list(name="site51")
)
results
```

```
   _Index_ Replication                          Site Start_Flag StartUp Cost StartDelay CountryDelay FirstPatient LastPatient
1        1          51     Chenango Memorial Hospital          1   22903 5500         77           37        22908       23320
2        2          51           Derry Medical Center          1   22895 4500         69           29        22899       23322
3        3          51        Lassen General Hospital          1   22902 6000         76           36        22906       23318
4        4          51          Sacred Heart Hospital          1   22895 5000         69           29        22899       23322
5        5          51 Shasta Regional Medical Center          1   22899 5500         73           33        22903       23322
6        6          51             St Eligus Hospital          1   22907 7500         81           41        22912       23323
7        7          51            Tower Medical Group          1   22898 6550         72           32        22904       23319
8        8          51      Twin Pines Medical Center          1   22899 7000         73           33        22905       23320
9        9          51      Western Regional Hospital          1   22897 8000         71           31        22901       23325
10      10          51          Wexler Medical Center          1   22897 4250         71           31        22901       23325
```

Display 5. Using SAS Code to subset table and read to R data frame

PYTHON AND SAS

For most SAS programmers, the main way to work with Python will be on the Viya platform. There are two main methods that will be used – writing Python programs that interact with SAS datasets and methods, and by using PROC PYTHON inside a SAS program. For using SAS code in Python, the SASpy library is used (check out the documentation in the recommended reading section).

The best way to see SAS and Python in action is by an example:

```
1
2   %* Define a SAS macro variable in SAS code;
3   %let language = 'python';
4
5   proc python;
6   submit;
7
8   print("Python in the SAS Log:")
9
10  # %*use symget to read a SAS macro variable into a python variable;
11  lang = SAS.symget('language')
12  ver = 3.8
13
14  # %* Submit SAS code inside python, using python syntax.  This dataset will live in WORK library
15  SAS.submit("data work.test; language={}; version={}; run;".format(lang,ver))
16
17
18  # %* Execute SAS functions with sasfnc;
19  var3 = SAS.sasfnc("upcase","hello world")
20  print( var3)
21
22  # %*Use symput to assign the value of a Python variable to a SAS macro;
23  py_var = 'Inside python'
24  SAS.symput('macrovar', py_var)
25
26
27  endsubmit;
28  run;
29
30  %* Show that the SAS macro variable persists and is populated;
31  %put &=macrovar;
32
33
34  %* Show that the test dataset created in the python code lives in SAS;
35  proc print data=test;
36  run;
```

Display 6. SAS and Python in the same program.

It's important to remember that the code inside the PROC Python code is straight-up Python code. The syntax formatting, however, is currently only beign applied to SAS keywords, which can be slightly misleading. Note that we had to start comments with a '**#**'. Also note that Python is case sensitive and that indentation means something specific (used in functions, loops and if..then statements), so be consistent with Python code formatting.

Let's examine what's happening in the lines of this program

Lines	Description
3	Assign a value to the SAS macro variable ***language***
5-6	Start the Python coding
8	Prints a message to the SAS log
11	Use the SASpy package to assign the value of ***&language*** to the python variable **lang**.
12	Python code for assigning a value to the variable **ver**
15	Use SASpy to submit text to the SAS engine. Note that the text inside the brackets is presented in Python format, with the .format() method being used to insert the values of **lang** and **ver** into the {} in the text string.
19	Assign a value to **var3** by applying a SAS function via the sasfnc method.
20	Print the value of **var3** to the log
24	Use the SAS.symput() method to assign the value of **py_var** to a SAS macro variable.
27-28	End the PROC PYTHON block
31	Put the value of ***&macrovar*** to the SAS log
35-36	Print the dataset that was created by the Python code in line 15

Table 2. Explanation of SAS/Python Code

Let's examine the log:

```
83   proc python;
84   submit
NOTE: Python initialized.
Python 3.8.5 (default, Sep  4 2020, 07:30:14)
[GCC 7.3.0] :: Anaconda, Inc. on linux
```

Display 7. SAS Log for lines 5-6

First up, you can see that Python was initialized, with the version number and source identified.

```
108  data work.test; language='python'; version=3.8; run;
NOTE: The data set WORK.TEST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

Display 8. SAS Log for line 15

Here you can see the translation of the Python code in the SAS.submit function from line 15. Note that it shows the actual code submitted (the way the MLOGIC & MPRINT options work in SAS). The notes indicate that the code was run and the TEST dataset was created.

```
>>>
Python in the SAS Log:
HELLO WORLD
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time               2.36 seconds
      cpu time                0.03 seconds
```

Display 9. SAS Log for lines 8 & 20

At the bottom of the log, you can see the output that was written in various lines in the program. It will all be placed together, so it is important to print blank lines for easier reading if applicable.

```
110  %* Show that the SAS macro variable persists and is populated;
111  %put &=macrovar;
MACROVAR=Inside python
```

Display 10. SAS Log for line 31

Finally, we see that a SAS macro variable was created in Python, and the variable and its value are available in the rest of the SAS program.

CONCLUSION

SAS has been integrated with open-source programming for years through integrating R in PROC IML, and the Viya environment opens up several new ways to code in a variety of language. For SAS programmers, PROC IML and PROC PYTHON will be most common ways to implement new methods. There are other ways open-source can be utilized that were beyond the scope of this paper. Some of those include:

- Open-source programming nodes in Viya Model Building
- API calls for R, Python and several other languages
- SWAT for Python

RECOMMENDED READING

SAS DOCUMENTATION

- PROC PYTHON documentation: https://go.documentation.sas.com/doc/en/pgmsascdc/v_017/proc/n0asd2rsj9aedgn1828aptww56of.htm
- R Haven Package: https://haven.tidyverse.org/
- SAS & Open Source Integration: https://www.sas.com/en_us/software/viya/open.html
- SAS Scripting Wrapper for Analytics Transfer (SWAT) for R: https://github.com/sassoftware/R-swat
- SASpy Library for Developers: https://developer.sas.com/guides/saspy.html

USER PAPERS

- Foreman, Carrie. SWAT's it all about? SAS Viya ® for Python Users. *SAS Global Forum 2019 Paper 3610-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3610-2019.pdf
- Lankham, Isiah & Slaughter, Matthew. Everything is better with friends: Executing SAS ® code in Phython scripts with SASpy. *SAS Global Forum 2019. Paper 3189-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3189-2019.pdf
- Matise, Joe. Connecting to Datasets through Python and SAS ®. MWSUG Paper 47-2019. Available at https://www.mwsug.org/proceedings/2019/AL/MWSUG-2019-AL-047.pdf
- Nakajima, Yuichi. Utilization of Python in clinical study by SASPy. *SAS Global Forum 2019 Paper 3191-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3191-2019.pdf
- Phillips, Jason. A Complete Introduction to SASPy and Jupyter Notebooks. *SAS Global Forum 2019 Paper 3238-2019*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3238-2019.pdf
- Varney, Brian. Getting Started with SAS ® Viya and the R SWAT package. *SESUG 2020 Paper 101*. Available at https://sesug.org/proceedings/sesug_2020_final_papers/Analytics_Leadership_and_Open_Analytics/SESUG2020_Paper_101_Final_PDF.pdf
- Vickery, John. Integrate Python with SAS ® using SASPy for a Simpler, More Effective Script. *SESUG Paper 152-2019*. Available at https://www.lexjansen.com/sesug/2019/SESUG2019_Paper-152_Final_PDF.pdf
- Weber, Tom. The History and Evolution of SASPy, Including an Overview of What It Can Do and How to Use It. *SAS Global Forum 2020 Paper SAS4141-2020*. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4141-2020.pdf

CONTACT INFORMATION

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

Jim Box
SAS Institute
Jim.box@sas.com

Samiul Haque
SAS Institute
Samiul.Haque@sas.com