PharmaSUG 2023 - Paper AD-079

Using the R interface in SAS ® to Call R Functions and Transfer Data

Bruce Gilsen, Federal Reserve Board

ABSTRACT

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.

This paper provides a basic introduction and some simple examples. The focus is on SAS users who are not PROC IML users, but who want to take advantage of the R interface.

INTRODUCTION

Starting in SAS ® 9.3, SAS users on Windows and Linux who license SAS/IML software can do the following:

- Call R functions from PROC IML.
- Transfer data between R and SAS:
  - Using SAS subroutines, you can transfer data between:
    - A SAS data set and an R data frame.
    - A SAS/IML matrix and an R matrix.
  - You can copy R data into the R interface with R functions like readRDS() and load(), and save R data from the R interface to a file with R functions like saveRDS(), save(), and save.image().

Two groups of potential users are as follows:

- SAS/IML users.
- All other SAS users, who can use PROC IML just as a wrapper to transfer data between SAS and R and call R functions.

The R interface allows you to call R from SAS, but not the reverse.

POSSIBLE USES

Possible uses of the R interface in SAS include the following:

- Use R data as input to a SAS program.
- Export SAS results to R for further processing.
- Use new econometric or statistical techniques:
  - R is easily extendible by users.
  - Shortly after publication in an econometric journal or elsewhere, new techniques are sometimes available in R that SAS requires a lengthier process (development, testing, etc.) to implement.
  - You can quickly and easily use the new technique in SAS by calling R instead of waiting for implementation in SAS.

SYSTEM REQUIREMENTS

System requirements for R interface in SAS include the following:

- You must execute SAS 9.3 or later on the Linux or Windows operating system.
- SAS/IML software must be installed. To determine if SAS/IML is installed, submit the following SAS statement: `proc setinit;run;`
- R must be installed. Wicklin (2013b) lists which R versions are supported by the SAS releases from 2009 – present.

CONFIGURING THE SAS INTERFACE TO R ON WINDOWS

Note: Configuration information could vary at your site. If necessary, contact local SAS support staff or SAS Technical Support for assistance.

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;
```

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;
```

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

```
-rlang
```

CONFIGURING THE SAS INTERFACE TO R ON LINUX

Note: Configuration information could vary at your site. If necessary, contact local SAS support staff or SAS Technical Support for assistance.

RLANG SYSTEM OPTION

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;
```

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

```
proc options option=config;run;
```

Linux SAS users typically have a combination of system and personal configuration files. For example, for SAS 9.4TS1M4 at the Federal Reserve, PROC OPTIONS shows four files: the first three are system files and the fourth is a personal file.

```
CONFIG=( /sftwr/sas/SASHome/SASFoundation/9.4/sasv9.cfg
/sftwr/sas/SASHome/SASFoundation/9.4/nls/en/sasv9.cfg
/sftwr/sas/SASHome/SASFoundation/9.4/sasv9_local.cfg
/it/support/it/m1bfg00/sasv9.cfg )
```

I recommend asking your network administrator to add –rlang in the /sftwr/sas/SASHome/SASFoundation/9.4/sasv9_local.cfg file.

If you cannot add –rlang to a configuration file, an alternative is to include it as an invocation option, as in the following commands:

- SAS Windowing environment: sas –rlang &

- Batch mode: sas –rlang myprog1.sas

R_HOME ENVIRONMENT VARIABLE

The R_HOME environment variable must be set to the R home directory, which is probably the value of the R_HOME_DIR environment variable in the script that invokes R (enter which R from a Linux prompt to locate the script). To determine if R_HOME is set in SAS, submit the following SAS statements:

```
data _null_;
  var1 = sysget("R_HOME");
  put var1=;
run;
```

If not set, ask your network administrator to add the following statement to the file bin/sasenv_local in the SASROOT directory (the primary directory containing SAS system software):

```
export R_HOME=name-of-R-home-directory
```

To determine the location of the SASROOT directory, submit the following SAS statement:

```
proc options option=config;run;
```

SYNTAX (INFORMAL)

Here is an informal description of the syntax of the R interface. For more details, see the chapter "Calling Functions in the R Language" in the "*SAS/IML 15.1: User's Guide*."

```
proc iml;
  (IML statements)
  call ExportDataSetToR("sasdataset", "R-frame" );
  call ExportMatrixToR(IML-matrix, "R-matrix");
  submit / R;       /* Start submitting statements to R */
    (R statements)
  endsubmit;        /* Stop submitting statements to R */
  call ImportDataSetFromR("sasdataset", "R-frame");
  call ImportMatrixFromR(IML-matrix, "R-matrix");
 (IML statements and submit blocks as necessary)
quit; /* end IML */
```

These commands do the following:

- PROC IML invokes SAS/IML software.
- The ExportDataSetToR subroutine copies a SAS data set to an R data frame.
- The ExportMatrixToR subroutine copies a SAS/IML matrix to an R matrix.
- The statements SUBMIT / R; and ENDSUBMIT; define the beginning and end of a submit block. All statements within a submit block are sent to R for execution. A SAS/IML session can include one or more submit blocks.
- The ImportDataSetFromR subroutine copies to a SAS data set an R data frame (or more generally, an R expression that can be coerced to an R data frame).
- The ImportMatrixFromR subroutine copies to a SAS/IML matrix an R matrix, data frame, or an R expression that can be coerced to an R data frame.

- Other IML statements can be included anywhere except within a submit block.
- QUIT; ends PROC IML.

SYNTAX (INFORMAL) FOR NON-SAS/IML USERS

If you are not a SAS/IML user, you can use PROC IML just as a wrapper to transfer data between SAS data sets and R data frames and call R functions, as follows:

```
proc iml;
  call ExportDataSetToR("sasdataset", "R-frame" );          /* optional */
  submit / R;       /* Start submitting statements to R */
    (R statements)
  endsubmit;        /* Stop submitting statements to R */
  call ImportDataSetFromR("sasdataset", "R-frame");         /* optional */
quit; /* end IML */
```

These commands do the following:

- The PROC IML, SUBMIT, ENDSUBMIT, and QUIT statements are always coded as shown.
- Optionally, ExportDataSetToR and ImportDataSetFromR transfer data between SAS data sets and R data frames.
- R statements are included in a submit block (between SUBMIT and ENDSUBMIT).

CURRENT WORKING DIRECTORY IN THE R ENVIRONMENT

Initially, the current working directory is the location of the temporary SAS WORK library.

You can determine the location of the WORK library with the following SAS code:

```
proc options option=work;run;
```

The WORK library is deleted at the end of your SAS session so it should not be used to store R data needed beyond the current SAS session.

You can modify the current working directory in a submit block with the R function setwd(), as in the code below. The current working directory persists across submit blocks until the SAS/IML session ends.

```
setwd("/u:/m1xxx00/sasandr")
```

PERSISTENCE IN THE R ENVIRONMENT

The R environment created in submit block(s) persists until the SAS/IML session ends, as follows:

- Data copied to or created in R in a submit block persists across submit blocks until the SAS/IML session ends.
- Plots created in R will continue to display until you close the plot window or end the SAS/IML session.
- The current working directory persists across submit blocks until the SAS/IML session ends.

To ensure that R data are preserved after the SAS/IML session ends, you can do one of the following:

- Outside of a submit block but before ending the SAS/IML session, use the ImportDataSetFromR or ImportMatrixFromR subroutines to copy R data to a SAS data set or IML matrix, as described above and illustrated in Examples 2, 3, and 4 below.
- In a submit block, save one or multiple R objects or an entire workspace to a file with the R functions saveRDS(), save(), or save.image(), as illustrated in Example 6 below.

SAVING THE ENTIRE WORKSPACE IN THE SAS INTERFACE TO R AND RESTORING IT IN R: CAUTION

The SAS interface to R defines the R function .Last() to prevent users from accidentally exiting R while it runs inside SAS/IML. If you save the entire workspace to a file in the SAS interface to R with the save.image() function and then restore it in R with the load() function, the .Last() function is saved and then restored, and when you try to end the R session with quit(), quit("no"), or quit("yes"), the following error occurs:

```
Error in .Last() : You cannot exit R when it is running inside SAS.
```

There are three ways to prevent this error:

- In the SAS interface to R, prior to using save.image(), remove the definition of .Last() with the following R statement, as illustrated in Example 6 below:

  ```
  rm(.Last)
  ```

- In R, after using the load() function to restore the workspace, remove the definition of .Last() with the following R statement:

  ```
  rm(.Last)
  ```

- When you terminate the R session, set the parameter "runLast" of the quit() function to FALSE:

  ```
  quit("no", runLast = FALSE)
  ```

WINDOWS FILE PATH LIMITATION

On Windows, using backslashes (\) in a file path generates an error. Use forward slashes (/) or double backslashes (\\) instead.

This generates an error:

```
pdf("u:\m1xxx00\sasandr\hist1.pdf")
```

Use one of the following instead:

```
pdf("u:/m1xxx00/sasandr/hist1.pdf")
```

```
pdf("u:\\m1xxx00\\sasandr\\hist1.pdf")
```

DATA TRANSFER DETAILS: MISSING VALUES, DATE/DATETIME/TIME VALUES, VARIABLE NAMES

The information in this section is illustrated in Example 2 below.

MISSING VALUES

When the ExportDataSetToR subroutine copies a SAS data set to an R data frame, it converts the standard SAS missing value of a single period (.) and the alternate missing values .A–.Z and ._ to the R missing value NA.

When the ImportDataSetFromR subroutine copies an R data frame to a SAS data set, it converts the R missing value NA to the standard SAS missing value of a single period (.).

For more information on missing and special values, see the *SAS/IML 15.1: User's Guide* (Calling Functions in the R Language chapter, Details of Data Transfer section, Special Numeric Values sub-section).

DATE/DATETIME/TIME VALUES

In SAS, date, time, and datetime values are numeric variables that represent data as follows:

- SAS date values are the number of days before or after January 1, 1960.

- SAS time values are the number of seconds since midnight on a given day.
- SAS datetime values are the number of seconds before or after January 1, 1960.

While not required, SAS date, time, and datetime values typically have a format associated with them from the family of date, time, or datetime formats, respectively, to facilitate meaningful display. For example, without a format, the SAS date value for May 8, 2018 is displayed as 21312, but you could associate a data format with the SAS date value such as DATE9. (08MAY2018 is displayed), YYMMDD8. (18-05-08 is displayed), or YYMMDDN8. (20180508 is displayed).

In R, classes are used to represent dates and datetimes. When copying a SAS data set to R, variables containing SAS date, time, and datetime values require an appropriate format to ensure proper transfer, because the format is used to determine the class of the variable in R, as follows.

Format family in SAS	Class in R
Date	Date
Time and Datetime	POSIXct and the pseudo-class POSIXt

For variables determined to be SAS time values, SAS converts them to datetime values whose date component is January 1, 1960.

When an R data frame is copied to SAS, the format of the variables is based on their class in R.

Class in R	Format in SAS
Date	DATE9.
POSIXt	DATETIME19.
All other cases	no format is assigned

For more information on date, datetime, and time values, see the *SAS/IML 15.1: User's Guide* (Calling Functions in the R Language chapter, Details of Data Transfer section, Date, Time, and Datetime Values sub-section).

VARIABLE NAMES

A valid R variable name would be invalid in SAS if it contains a period (.) or is longer than 32 characters. When the ImportDataSetFromR subroutine copies an R data frame to a SAS data set, it converts invalid R variable names to valid SAS variable names as follows:

- Periods are converted to underscores.
- Names longer than 32 characters are truncated to 32 characters.
- Any duplicate variable names generated by the above steps are converted to unique names by appending a number and (if necessary to limit the length to 32 characters) truncating the name prior to appending.

EXAMPLE 1: SIMPLE EXAMPLE OF CALLING R FUNCTION WITH NO DATA TRANSFER

This is a simple example with no data transfer between SAS and R. We take the mean of a variable in SAS, then in R, and compare the results:

```
  /* Create data, take mean in SAS */
data one;
  input x;
  datalines;
2
```

```
4
6
9
11
100
;run;
proc means data=one mean;
  var x;
run;
  /* Create data, take mean in R */
proc iml;
  submit / R;
    x<-c(2,4,6,9,11,100)
    meanx<-mean(x)
    print(meanx)
  endsubmit;
quit; /* end IML */
```

The results are the same in SAS and R. The following output is displayed:

```
                      The MEANS Procedure       <=== SAS PROC MEANS output

                     Analysis Variable : x

                                Mean
                     ------------
                       22.0000000
                     ------------

[1] 22                                                    <=== R output
```

Output 1. Output From Example 1

EXAMPLE 2: COPY A SAS DATA SET TO AND FROM AN R DATA FRAME

In this example, we do the following:

- Create a SAS data set that includes a missing value and a SAS date.
- Copy the SAS data set to an R data frame with the ExportDataSetToR subroutine.
- Add a column to the data frame whose name is an invalid SAS variable name because it contains a character not valid in SAS (.).
- Display data frame information: variable names and values, column value, SAS date variable class, data frame structure, and data summary.
- Copy the R data frame to a SAS data set with the ImportDataSetFromR subroutine and display the values.

Note the following in the results:

- ExportDataSetToR converts the standard SAS missing value of a single period (.) to the R missing value NA.
- DATE1, which is a SAS date, has a class of DATE in R.
- ImportDataSetFromR does the following when it creates SAS data set FROMIMLTWO:
  - Convert the R missing value NA to the standard SAS missing value of a single period (.).
  - Convert invalid SAS variable name net.income to valid SAS variable name net_income by changing the period to an underscore.
  - Assign DATE1 a format of DATE9 because DATE1 has the class of DATE in R.

```
data two;         /* Part I: Create data set in SAS */
  input tax income date1 yymmdd8.;
  format date1 yymmddn8.;
  datalines;
1 2 20180301
3 4 20180401
5 6 20180501
7 8 20180601
. 10 20180701
;run;
proc iml;         /* Part II: Call R in PROC IML */
    /* Copy SAS data set TWO (WORK.TWO) to R data frame IMLTWO. */
  call ExportDataSetToR("two", "imltwo");
   /* Display data frame information: variable names and values,
      a column value, class of the SAS date variable,
      data frame structure, and data summary */
  submit / R;
    imltwo$net.income<-imltwo$income-imltwo$tax
    names(imltwo)
    print(imltwo)
    print(imltwo$income)
    class(imltwo$date1)
    str(imltwo)
    summary(imltwo)
  endsubmit;
    /* Part III: Copy R data frame IMLTWO to SAS data set FROMIMLTWO
       (WORK.FROMIMLTWO), end PROC IML, print values. */
  call ImportDataSetFromR("fromimltwo", "imltwo");
quit;  /* end PROC IML */
```

```
proc print data=fromimltwo;
run;
```

The following output is displayed:

```
                                                    <=== output of names(imltwo)
[1] "tax"        "income"     "date1"      "net.income"
                                                    <=== output of print(imltwo)
  tax income      date1 net.income

1   1      2 2018-03-01          1
2   3      4 2018-04-01          1
3   5      6 2018-05-01          1
4   7      8 2018-06-01          1
5  NA     10 2018-07-01         NA
                                          <=== output of print(imltwo$income)
[1]  2  4  6  8 10
                                           <=== output of class(imltwo$date1)
 [1] "Date"
                                                     <=== output of str(imltwo)
'data.frame':   5 obs. of  4 variables:
 $ tax       : num  1 3 5 7 NA
 $ income    : num  2 4 6 8 10
 $ date1     : Date, format: "2018-03-01" "2018-04-01" ...
 $ net.income: num  1 1 1 1 NA
                                                 <=== output of summary(imltwo)
      tax          income       date1              net.income
 Min.   :1.0   Min.   : 2   Min.   :2018-03-01   Min.   :1
 1st Qu.:2.5   1st Qu.: 4   1st Qu.:2018-04-01   1st Qu.:1
 Median :4.0   Median : 6   Median :2018-05-01   Median :1
 Mean   :4.0   Mean   : 6   Mean   :2018-05-01   Mean   :1
 3rd Qu.:5.5   3rd Qu.: 8   3rd Qu.:2018-06-01   3rd Qu.:1
 Max.   :7.0   Max.   :10   Max.   :2018-07-01   Max.   :1
 NA's   :1                                       NA's   :1
                                                 <=== output of proc print
                                      net_
   Obs    tax    income        date1    income

    1      1        2       01MAR2018       1
    2      3        4       01APR2018       1
    3      5        6       01MAY2018       1
    4      7        8       01JUN2018       1
    5      .       10       01JUL2018       .
```

Output 2. Output from Example 2

EXAMPLE 3: RUN REGRESSION IN R, RETURN RESULTS TO SAS

In this example, we do the following:

- Create a SAS data set and copy it to an R data frame.
- Run a regression in SAS with PROC REG and then in R to compare the results.
- Return the regression results from R to SAS and calculate predicted values using parameter estimates generated in R.

```
data Class;       /* Create a data set in SAS */
  length Name $8;
  input Name $ Height Weight;
  datalines;
Alfred   69.0 112.5
Alice    56.5  84.0
Barbara  65.3  98.0
Carol    62.8 102.5
Henry    63.5 102.5
James    57.3  83.0
Jane     59.8  84.5
Janet    62.5 112.5
Jeffrey  62.5  84.0
John     59.0  99.5
Joyce    51.3  50.5
Judy     64.3  90.0
Louise   56.3  77.0
Mary     66.5 112.0
Philip   72.0 150.0
Robert   64.8 128.0
Ronald   67.0 133.0
Thomas   57.5  85.0
William  66.5 112.0
;run;

proc reg data=Class;     /* Do OLS regression in SAS as a comparison */
  model Weight = Height;
run;

proc iml;

    /* Copy SAS data set Class to R data frame Class1 */
  call ExportDataSetToR("Class", "Class1");

  submit / R;  /* Begin R submit block */

      # Do regression in R
    Model <- lm(Weight ~ Height, data=Class1, na.action="na.exclude")
      # Create and display R data objects with coefficients,
      # fitted values, residuals
    ParamEst <- data.frame(Intercept = coef(Model)[1],
        Height_Parameter = coef(Model)[2])
    Pred     <- fitted(Model)
    Resid    <- residuals(Model)
    print(ParamEst)
    print(Pred)
  endsubmit;  /* End R submit block, return to SAS/IML */

    /* Retrieve parameter estimates from R */
  call ImportDataSetFromR("pe", "ParamEst");

quit; /* end PROC IML */

  /* Display parameter estimates from R */
title Parameter estimates copied from R to SAS ;
proc print data=pe;
run; title;
```

```
  /* Use parameter estimates to compute
     predicted values for some heights */
data predictweightfromheight;
  set pe;
  keep height predicted_weight;
  do height = 55 to 70 by 5;
    predicted_weight = intercept + height * Height_Parameter;
    output;
  end;
run;
title Predicted weight from height;
proc print data=predictweightfromheight;
run;
```


The following output is displayed:

```
                                                <=== SAS PROC REG output
                              The REG Procedure
                                 Model: MODEL1
                           Dependent Variable: Weight

                        Number of Observations Read          19
                        Number of Observations Used          19

                                 Analysis of Variance

                                       Sum of           Mean
   Source                 DF         Squares         Square    F Value    Pr > F

   Model                   1      7193.24912     7193.24912      57.08    <.0001
   Error                  17      2142.48772      126.02869
   Corrected Total        18      9335.73684

                  Root MSE             11.22625    R-Square     0.7705
                  Dependent Mean      100.02632    Adj R-Sq     0.7570
                  Coeff Var            11.22330

                                 Parameter Estimates

                                 Parameter       Standard
          Variable     DF         Estimate          Error    t Value    Pr > |t|

          Intercept     1     -143.02692       32.27459      -4.43      0.0004
          Height        1       3.89903        0.51609       7.55      <.0001

                                                    <=== R output starts here
              Intercept Height_Parameter      <=== output of print(ParamEst)
(Intercept) -143.0269          3.89903
```

```
                                                  <=== output of print(Pred)
        1         2         3         4         5
126.00617  77.26829 111.57976 101.83218 104.56150
        6         7         8         9        10
 80.38752  90.13509 100.66247 100.66247  87.01587
       11        12        13        14        15
 56.99333 107.68073  76.48849 116.25859 137.70326
       16        17        18        19
109.63024 118.20811  81.16732 116.25859

                                              <=== SAS output starts here

                  Parameter estimates copied from R to SAS

                                          Height_
                       Obs    Intercept    Parameter

                        1      -143.027     3.89903

Predicted weight from height

                                        predicted_
                        Obs    height      weight

                         1       55         71.420
                         2       60         90.915
                         3       65        110.410
                         4       70        129.905
```

Output 3. Output from Example 3

EXAMPLE 4: COPY AN IML MATRIX TO R, CALL AN R PACKAGE AND R GRAPHICS

In this example, we do the following:

- Create an IML vector and copy it to R.
- Pass a parameter to R.
- Use the R LIBRARY function to load an R package.
- Call R functions to analyze the data.
- Copy the results of the R analysis to SAS/IML vectors and analyze them in SAS.
- Call R to display a plot. The plot window stays open until we delete it or end SAS/IML with the QUIT; statement.
- Call R to create a PDF file containing the plot. As noted above, specifying a file path on Windows with backslashes (\) generates an error. Use forward slashes (/) or double backslashes (\\) instead.

```
  /*I. Create IML vector q, transfer it to R as vector rq */
proc iml;
  q = {3.7, 7.1, 2, 4.2, 5.3, 6.4, 8, 5.7, 3.1, 6.1, 4.4, 5.4, 9.5, 11.2};
  RVar = "rq";
  call ExportMatrixToR( q, RVar );

    /* II. Pass name of matrix to R as parameter.  Load
       KernSmooth package, compute kernel density estimate. */
```

```
  submit RVar / R;
    library(KernSmooth)
    idx <-which(!is.na(&RVar))        # exclude missing values
    p <- &RVar[idx]                 # from KernSmooth functions
    h = dpik(p)             # Sheather-Jones plug-in bandwidth
    est <- bkde(p, bandwidth=h)            # est has 2 columns
  endsubmit;

    /* III Copy results to IML matrix, do more computations */
  call ImportMatrixFromR( m, "est" );
                             /* estimate density for q >= 8 */
  x = m[,1];                        /* x values for density */
  idx = loc( x>=8 );                  /* find values x >= 8 */
  y = m[idx, 2];    /* extract corresponding density values */

    /* Use trapezoidal rule to estimate area under density curve.   Area
       of trapezoid with base w and heights h1 and h2 is w*(h1+h2)/2. */
  w = m[2,1] - m[1,1];
  h1 = y[1:nrow(y)-1];
  h2 = y[2:nrow(y)];
  Area = w * sum(h1+h2) / 2;
  print Area;

    /* IV. Display plot, then save it to a PDF file. Displayed file
       persists until we delete it or end SAS/IML with the QUIT; statement.
       For Windows file name path use a double "\\" or "/" instead
       of "\" in the path to avoid an error. */
  submit / R;
    hist(p, freq=FALSE) # display histogram
    lines(est)          # kde overlay

      # Code for Linux
    pdf("/my/home/m1xxx00/sasandr/hist1.pdf")
      # Code for Windows, use "/", not "\" in file path
    # pdf("u:/m1xxx00/sasandr/hist1.pdf")

    hist(p, freq=FALSE) # write histogram to PDF file
    lines(est)          # kde overlay
    dev.off()
  endsubmit;
quit; /* end PROC IML */
```


The following output is displayed by the print Area statement in section III:

```
                Area
           0.2118117
```

Output 4. Output from Example 4

EXAMPLE 5: PROCESS R STATEMENTS IN A SOURCE FILE

The source function causes R to read and process the R statements contained in a file. In this example, Linux file /my/home/directory/rprogs/rcode1.R and Windows file u:/m1xxx00/rprogs/rcode1.R are source files:

```
proc iml;
  /* optionally copy data from SAS to R */

  submit / R;
      # Code for Linux
    source("/my/home/directory/rprogs/rcode1.R")
      # Code for Windows, use "/", not "\" in file path
    # source("u:/m1xxx00/rprogs/rcode1.R")
  endsubmit;

     /* optionally copy data from R to SAS */

quit; /* end IML */
```

EXAMPLE 6: SAVE OR READ ONE OR MULTIPLE R OBJECTS OR AN ENTIRE WORKSPACE TO/FROM A FILE

First, in R, do the following:

- Change the current working directory with setwd().
- Create two objects (variables) and a data frame.
- Save R data to different files as follows:
  - A single object (variable), object1, to a file with the saveRDS() function.
  - Two objects (variables), object1 and object2, to a file with the save() function.
  - The entire workspace to a file with the save.image() function. (Note: not recommended if you have very large data files or have created a lot of intermediate objects in your current R session.)

```
setwd("~m1xxx00/rstuff")              # set current working directory
object1<-1:5                          # create an object, assign value
object2<-11:15                        # create 2nd object, assign value

                                      # create data.frame w/2 objects,
                                      #   assign each a new name
df<- data.frame(column1=object1, column2=object2)

                                      # save object1 to external file
saveRDS(object1, file = "mydata_object1.rds")
object1[3:5] = -1*object1[3:5]        # modify 3rd-5th values of object1

                                      # save 2 objects to external file
save(object1, object2, file = "objects_1_and_2.RData")

                                      # save entire workspace, could
                                      #   contain unnecessary objects so
                                      #   do with caution
save.image(file = "my_entire_work_space1.RData")

                                      # exit R session without saving
                                      #   current workspace to a .Rdata
```

```
quit("no")                                #   in current working directory
```

Then, in SAS, read each of the saved objects, modify the data a bit, and save R data to files with each of the same three functions:

```
proc iml;
  submit / R;
  setwd("~m1xxx00/rstuff")                # set current working directory

                                            # read, print object
  object1_from_r= readRDS(file = "mydata_object1.rds")
  print(object1_from_r)

  load("objects_1_and_2.RData")           # read, print 2 objects
  print(object1)
  print(object2)

                                          # restore workspace (do with caution)
  load(file = "my_entire_work_space1.RData")
  print(df)                               #   if large), print dataframe

  library(dplyr)                          # need for mutate() to work

                                          # add column to data frame, print
  df <- mutate(df, column3 = ifelse(column2>12, log(column2), column2))
  print(df)

                                          # save object1_from_r to new
                                          # external file
  saveRDS(object1_from_r, file = "mydata_object1_from_r.rds")

                                          # save 2 objects to external file,
                                          #   overwrite previous file
  save(object1, object2, file = "objects_1_and_2.RData")

                                          # save entire workspace, overwrite
                                          # previous file.  Workspace
                                          # available in R and in future
                                          # SAS/R sessions.  To prevent error
                                          # in R, remove definition of .Last
                                          # function as explained above.
  rm(.Last)
  save.image(file = "my_entire_work_space1.RData")

  endsubmit;
quit;
```

The following output is displayed:

```
 [1] 1 2 3 4 5                                <=== print(object1_from_r)
 [1]  1  2 -3 -4 -5                           <=== print(object1)
 [1] 11 12 13 14 15                           <=== print(object2)
   column1 column2                            <=== print(df)
1       1      11
2       2      12
3       3      13
```

```
4        4       14
5        5       15
  column1 column2   column3              <=== print(df) after mutate()
1       1       11 11.000000
2       2       12 12.000000
3       3       13  2.564949
4       4       14  2.639057
5       5       15  2.708050
```

Output 5. Output from Example 6

CONCLUSION

This paper provided an introduction to the SAS Interface to R, which allows SAS users to easily call R functions and transfer data between SAS and R from within SAS. This could prove to be a very useful tool for SAS users who have SAS/IML software installed.

REFERENCES

SAS Institute Inc. (2018), "SAS/IML® 15.1 User's Guide," Cary, NC: SAS Institute Inc.

Wicklin, Rick (2013a) "Twelve advantages to calling R from the SAS/IML language," The DO Loop blog, published 25NOV2013. https://blogs.sas.com/content/iml/2013/11/25/twelve-advantages-to-calling-r-from-the-sasiml-language.html

Wicklin, Rick (2013b) "What versions of R are supported by SAS?," The DO Loop blog, published 16SEP2013 and updated through February 2021. < https://blogs.sas.com/content/iml/2013/09/16/what-versions-of-r-are-supported-by-sas.htmll>

ACKNOWLEDGMENTS

Support from the following people is greatly appreciated. Rob Agnelli at SAS Institute provided technical support on several occasions. Rick Wicklin at SAS Institute provided feedback on an early draft of this paper. William Ampeh, Donna Hill, Heidi Markovitz, and Peter Sorock at the Federal Reserve Board all contributed to the development of this paper. The code in examples 3 and 4 are closely based on examples in the "SAS/IML® 15.1 User's Guide."

CONTACT INFORMATION

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

Bruce Gilsen
bruce.gilsen@frb.gov