SESUG Paper 104-2019

%SUBMIT_R: A SAS$^{(R)}$ Macro to Interface SAS and R

Ross Bettinger, Consultant

ABSTRACT

The purpose of the %SUBMIT_R macro is to facilitate communication between SAS® and R under Windows. %SUBMIT_R uses SAS' unnamed pipe device type to invoke the R executable. SAS datasets may be converted into R data frames and vice versa in a manner similar to using the SAS/IML ExportDataSetToR and ImportDataSetFromR functions. R graphics are also supported, and are displayed in the SAS results viewer. Graphs may be saved in user-specified locations as image files. R scripts may be created using a SAS data _null_ step to write a file containing an R script, read from a user-specified .R input file, or by using the %R macro. Output of R execution may be directed to the SAS log file for inspection or to a user-specified .Rout file for later use.

Keywords: SAS macro, R script, SAS program file, unnamed pipe, ODS, HTML

INTRODUCTION

The features of the %SUBMIT_R macro provide a connection between SAS and R so that users have access to SAS' data handling capabilities to process large volumes of data efficiently and then transmit the results to R for further manipulation. For example, you can use the %SUBMIT_R macro to leverage SAS' strengths in data handling to prepare data for analysis and then use R to produce graphics, or to use state-of-the-art machine learning techniques.

The %SUBMIT_R macro uses a previously-defined R script in its execution. This script may be created within the body of a SAS program using the %R macro or stored in a file independent of SAS and supplied via a macro parameter. Since it is a standard R script, it may contain any statement accepted by the R interpreter.

SAS datasets may be transmitted to the R environment and represented as data frames via the ExportDataSetToR function. Reciprocally, R data frames may be formatted as SAS datasets and stored in the temporary SAS work directory or in a permanent SAS library using the ImportDataSetFromR function. These functions are patterned after their eponymous SAS/IML counterparts but are used within the R script instead of bracketing the SAS/IML submit/R code block. SAS/IML is not required to use the %SUBMIT_R macro.

R graphics may be created in your R script and are displayed in the SAS Results Viewer window. If you specify a destination other than the default temporary SAS work directory, the image files that you create will be stored in that directory.

MACRO PARAMETERS

The %SUBMIT_R macro declaration and description of parameters is shown below.

```
%macro SUBMIT_R(
      RSCRIPT= /* [optional] name of file containing R statements                 */
    , RLOG=    /* [optional] name of file containing results of running &SUBMIT_R */
    , RCMD=    /* [optional] path to R executable                                 */
    , ROPTS=   /* [optional] list of option(s) to use when executing R in batch   */
    , SETWD=   /* [optional] R working directory path                             */
    , GPATH=   /* [optional] location for all graphics output that is generated   */
    ) ;

    /* purpose: submit R script for execution, retrieve results as log file
     *
     * parameters:
     *  RSCRIPT ::= name of text file containing R statements
     *              if &RSCRIPT is null, take statements from global var &R\_SCRIPT
     *
     *  RLOG    ::= name of text file containing results of &RSCRIPT execution
     *              if &RLOG is null put results to SAS log
     *
     *  RCMD    ::= path to R executable.
     *              if &RCMD is null, default option is defined below
     *
     *  ROPTS   ::= list of R command line options
     *              if &ROPTS is null, default options are --quiet --vanilla
     *
     *  SETWD   ::= R environment working directory
     *              if &SETWD is null, default value is &SAS_TEMP
     *
     *  GPATH   ::= directory in which all graphical output produced by R is stored
     *              if omitted, ODS puts graphics into the SAS Temporary Files dir
     */
```

The phrase "if `&MACROVAR` is null" requires explanation. A SAS macro parameter has a length property: if it is not defined, i.e., given a value at macro invocation, it has a null value and hence a length of 0. Thus, if the macro parameter `&RSCRIPT` is not set to a value (see the Examples of Use, below), its contents are null, it has a length of 0. As an example, the nullity of `&RSCRIPT` will be tested and if the predicate is true, R script statements will be taken from the global macro variable `&R_SCRIPT` that is created by the `%R` macro.

CREATING AN R SCRIPT WITHIN A SAS PROGRAM

The `%R` macro manages creation of an R script within the body of a SAS program file. It must be invoked prior to invoking the `%SUBMIT_R` macro because it produces a global macro variable (called `&R_SCRIPT`) that is visible to the `%SUBMIT_R` macro. The `&R_SCRIPT` macro variable contains R statements to be executed by the R executive program.

The following code snippet illustrates the use of the `%R` macro to build an R script for use by the `%SUBMIT_R` macro. Note that since no parameters were supplied to the `%SUBMIT_R` macro, default values will be used.

```
%R(
    f1 <- 1 ; f2 <- 1 ; f3 <- f1 + f2 ;
    f4 <- f2 + f3 ;
    f5 <- f3 + f4 ;
    cat( "The first five Fibonacci numbers are", c( f1, f2, f3, f4, f5 ), "\n" )
  )

%SUBMIT_R
```

The semicolon (;) line separator is used to delimit the end of an R statement so that each one is a distinct R statement. Since the default values are used, the results were sent to the SAS log, and the `setwd` command was used to set the working directory to the SAS Temporary Files path, which is the default location.

This script produces the following output:

```
/******************************** Begin R Output ******************************\
> setwd( "C:/Users/userid/AppData/Local/Temp/SAS Temporary Files/Tempdir" )
> f1 <- 1
> f2 <- 1
> f3 <- f1 + f2
> f4 <- f2 + f3
> f5 <- f3 + f4
> cat( "The first five Fibonacci numbers are", c( f1, f2, f3, f4, f5 ), "\n" )
The first five Fibonacci numbers are 1 1 2 3 5
\******************************** End R Output ********************************/
```

EXAMPLES OF USE

Herein are described several examples of using the `%SUBMIT_R` macro to indicate its use with and without the `%R` script generator macro and to show the use of other features, e.g., SAS-to-R dataset/data frame transfer, graphics generation, and `%R` script creation.

EXAMPLE: “HELLO, WORLD!”

Features:

- Using the `%R` macro to create an R script
- Directing R output to the SAS log

SAS Program

```
%R( print( "Hello, World!" ))

%SUBMIT_R
```

The R output written to the SAS log is shown below. It contains the same results as if the R statements were executed directly by the R executive.

```
/******************************** Begin R Output *****************************\
> setwd( "C:/Users/userid/AppData/Local/Temp/SAS Temporary Files/Tempdir" )
> print( "Hello, World!" )
[1] "Hello, World!"
\******************************** End R Output *******************************/
```

EXAMPLE: GRAPHS OF ARCHIMEDEAN SPIRAL

Features:

- Exporting a SAS dataset to an R data frame
- Using the `%R` macro to create an R script
- Displaying R graphical output in the SAS Result Viewer

The Archimedean spiral is a graph of the polar equation $r = a\theta^{\frac{1}{n}}$ where r is the radial distance, θ is the polar angle, and n is a constant of curvature [Weisstein(2014a)]. In this example, $a = 1$, $n = 1$, and $\theta = (0)(\pi/50)(2\pi)$.

This example uses the `ExportDataSetToR` command. The `%SUBMIT_R` macro parses the command and reads the `work.spiral` dataset that was created in the SAS Temporary Files directory. It is written as a CSV file into the R working directory where it is converted to the data frame `spiral`. The four curves in the `spiral` data frame are then plotted and saved in the `Spiral.png` file in the R working directory (which will be the SAS Temporary Files directory by default).

SAS Program

```
/* create Archimedean spiral  */

data spiral ;
    phase_inc = constant( 'pi' ) / 2 ;

    do theta = 0 to 2 * constant( 'pi' ) by constant( 'pi' ) / 50 ;
        x0 = theta * cos( theta ) ;
        y0 = theta * sin( theta ) ;

        x1 = theta * cos( theta + phase_inc ) ;
        y1 = theta * sin( theta + phase_inc  ) ;

        x2 = theta * cos( theta + 2 * phase_inc  ) ;
        y2 = theta * sin( theta + 2 * phase_inc  ) ;

        x3 = theta * cos( theta + 3 * phase_inc  ) ;
        y3 = theta * sin( theta + 3 * phase_inc  ) ;

        output ;
    end ;
```

```
run ;

%R(
    library( grDevices ) ;

    # use the ExportDataSetToR command to produce a CSV file  ;
    # and %SUBMIT_R will embed the R read.csv statement to convert it into a data frame ;

    ExportDataSetToR( work.spiral, spiral ) ;

    # open the file Spiral.png to receive graphics output ;

    png( file="Spiral.png", bg="transparent" ) ;

    # define the min, max boundaries of the plot ;

    min_max <- c( -2 * pi, 2 * pi ) ;

    # and produce overlaid plots in the Spiral.png file ;

    plot( spiral$x0, spiral$y0
        , type="p", main="Archimedean Spiral"
        , xlim=min_max, ylim=min_max
        , xlab="r cos(theta)", ylab="r sin(theta)"
        , col="red"
        ) ;

    points( spiral$x1, spiral$y1, col="green", xlim=min_max, ylim=min_max ) ;
    points( spiral$x2, spiral$y2, col="blue"  , xlim=min_max, ylim=min_max ) ;
    points( spiral$x3, spiral$y3, col="black" , xlim=min_max, ylim=min_max ) ;

    # close the graphics device file ;

    dev.off() ;
)

%SUBMIT_R
```

The graph is shown below.

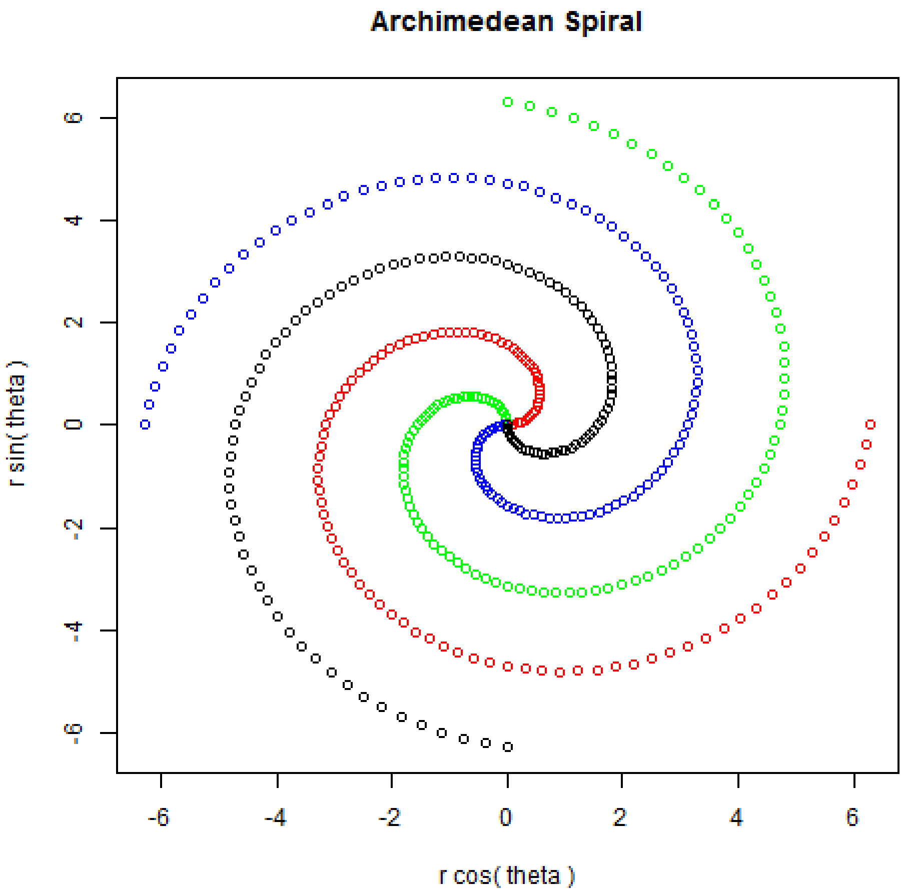

Figure 1: Archimedean Spiral

EXAMPLE: MATRIX COMPUTATION OF FIBONACCI SERIES

Features:

- Using `%R` macro to create an R script
- Importing a SAS dataset from an R vector using the `ImportDataSetFromR` function
- Saving the `%R` output into a Windows file

The Fibonacci series is defined by the recursive relationship $F_n = F_{n-1} + F_{n-2}$ for which $F_1 = 1$ and $F_2 = 1$. The series may be computed iteratively or as the solution to a system of linear equations, as represented by the R script below.

SAS Program

```
%R(
    f_mat <- matrix( c(  1,   0,   0,   0,   0,   0, 0,
                         0,   1,   0,   0,   0,   0, 0,
                        -1,  -1,   1,   0,   0,   0, 0,
                         0,  -1,  -1,   1,   0,   0, 0,
                         0,   0,  -1,  -1,   1,   0, 0,
                         0,   0,   0,  -1,  -1,   1, 0,
                         0,   0,   0,   0,  -1,  -1, 1
                         )
                   , nrow=7, byrow=T
                   ) ;
    f_next   <-  matrix( c( 1, 1, 0, 0, 0, 0, 0 ), nrow=7, ncol=1 ) ;
    f_series <- solve( f_mat, f_next ) ;
    cat( "f_series =", f_series, "\n" ) ;

    ImportDataSetFromR( Fibonacci_Series, f_series )
)

%SUBMIT_R( RLOG=C:/Users/userid/Documents/My SAS Files/R/Matrix_Fibonacci.Rout )
```

produces the result

```
f_series = 1 1 2 3 5 8 13
```

in the R workspace and stores the values in SAS dataset `work.Fibonacci_Series`. The .Rout file is saved in the user-specified location.

EXAMPLE: RECURSIVE COMPUTATION OF PI

Features:

- Using `%R` macro embedded within a SAS macro to create an R script

Vieta formulated an expression in 1593 for π as the product of an infinite number of nested radicals [Weisstein(2014b)]. The formula is

$$\frac{2}{\pi} = \lim_{n\to\infty} \prod_{i=1}^{n} a_i = \sqrt{\frac{1}{2}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}}} \cdots$$

with the initial condition $a_1 = \sqrt{\frac{1}{2}}$ and recurrence relation $a_i = \sqrt{\frac{1}{2} + \frac{1}{2}a_{i-1}}$.

The accuracy of the approximation depends on the number of terms used in the nested radical expansion. In this case, we use a SAS macro to assemble up to n terms.

SAS Program

```
%macro VIETA( N ) ;
    /* purpose: use %R to build recursive R function that uses Vieta's formula */

    %R(
        Vieta <- function( n )
        {
            if( n == 0 )
                return( sqrt( 1/2 ))
            else
                return( sqrt( 1/2 + 1/2 * Vieta( n-1 )))
        } ;

    /* use SAS macro code to produce repeated computations
     * note use of semicolon (;) line delimiter to separate R statements in body of macro
     */

    %do I = 1 %to &N ;

        two_over_pi <- Vieta( 0 ) ;
        for( n in 1 : &I ) two_over_pi <- two_over_pi * Vieta( n ) ;
            Vieta_pi <- 2 / two_over_pi ;
        cat( "Pi computed by Vieta's formula =", format( Vieta_pi, digits=10 ), "\n" ) ;

    %end ;
    )
```

```
%mend VIETA ;

%VIETA( 10 )

%SUBMIT_R
```

The SAS macro `%VIETA` assembles R statements within the `%R` macro. The `%R` macro then creates a global macro variable that is used by `%SUBMIT_R` and sent to the R executive for processing. The last execution of the R script returns the following (abridged) output. Note that the macro parameter `&N` is substituted for `&I` in the `for` loop.

```
/******************************* Begin R Output ******************************\
> two_over_pi <- Vieta( 0 )
> for( n in 1 : 10 ) two_over_pi <- two_over_pi * Vieta( n )
> Vieta_pi <- 2 / two_over_pi
> cat( "Pi computed by Vieta's formula =", format( Vieta_pi, digits=10 ), "\n" )
Pi computed by Vieta's nested radical formula = 3.141592346
\******************************* End R Output *******************************/
```

EXAMPLE: COMPUTING PYTHAGOREAN TRIPLETS

Features:

- Creating a user-defined R script within a SAS program
- Transmitting parameter values to an R program using the SAS macro language
- Reading R statements from a user-specified file
- Displaying R graphical output in the SAS Result Viewer
- Importing an R data frame to a SAS dataset

A Pythagorean triplet is a triplet (a, b, c) of the relationship $a^2 + b^2 = c^2$ where c^2 is a perfect square, e.g., c is an integer. In this example, we compute all $m \cdot n$ combinations of the Cartesian product $outer((1 : m)^2, (1 : n)^2, ''+'')$, select the Pythagorean triplets, and create 2D and 3D scatterplots of (a, b, c). The values of m and n are set by SAS macro variables `&M` and `&N` in the SAS program, and the R script is written to a file called `Triplets.R`. The triplets are imported from the R environment to the permanent SAS dataset `Pythagorean_Triplets` in SAS library `Triplets`.

SAS Program

```
/* use SAS macro vars to transmit values to R function */

%let M = 50 ; %let N = 50 ;

%let SAS_TEMP = %sysfunc( translate( %sysfunc( pathname( work )), %str(/), %str(\))) ;

libname TRIPLETS '/C:/Users/userid/Documents/My SAS Files/Triplets' ;

filename triplets "&SAS_TEMP/triplets.R" ;

data _null_ ;
    file triplets ;
    put "library(grDevices) ;" ;
    put "triplets <- function( m, n ) " ;
    put "# compute Pythagorean triplets " ;
    put "   a2 <- ( 1 : m )^ 2 ;" ;
    put "{ " ;
    put "  b2 <- ( 1 : n )^ 2 ;" ;
    put "  a2_b2 <- outer( a2, b2, ""+"" ) ;" ;
    put "  c <- ( sqrt( a2_b2 ) %% 1) == 0 ;" ;
    put "  indices <- matrix( 0, m*n, 3 ) ;" ;
    put " for( i in 1 : m ) " ;
    put " for( j in 1 : n ) " ;
    put "    if( c[ i, j ] ) indices[ i, ] <- c( i, j, sqrt( a2_b2[ i, j ] )) ;" ;
    put "  triplets <- indices[ indices[ , 1 ] > 0, ] ;" ;
    put "} ;" ;
    put " " ;
    put "# SAS macro vars supply values to R function via substitution ;" ;
    put "# m = &M, n = &N ;" ;
    put "trio <- triplets( &M, &N ) ;" ;
    put "print( trio ) ;" ;
    put "png(file=``Triplets.png'', bg=``transparent'') ;" ;
    put "png( filename=""C:/Users/userid/Documents/My SAS Files/R/Triplets_2D.png"",
         bg=""transparent"") ;" ;
    put "plot( trio[,1], trio[,2], main=""Pythagorean Triplets 2d"", xlab=""a"", ylab=""b"",
         pch="" "" ) ;" ;
    put "s2d.coords <- xy.coords(trio[,c(1,2)]) ;" ;
    put "text( s2d.coords$x,s2d.coords$y, cex=1, labels=as.character(trio[,3])) ;" ;
    put "dev.off() ;" ;
    put "png( filename=""C:/Users/userid/Documents/My SAS Files/R/Triplets_3D.png"",
         bg=""transparent"") ;" ;
    put "s3d <- scatterplot3d( trio, xlab=""a"", ylab=""b"", zlab=""c"",
         main=""Pythagorean Triplets 3D"", pch="" "", type=""p"" ) ;" ;
    put "s3d.coords <- s3d$xyz.convert(trio) ;" ;
    put "text( s3d.coords$x, s3d.coords$y, s3d.coords$z, cex=1,labels=as.character(trio[,3])) ;" ;
    put "dev.off() ;" ;
    put "ImportDataSetFromR( TRIPLETS.Pythagorean_Triplets, trio ) ;" ;
run ;

%SUBMIT_R( RSCRIPT=&SAS_TEMP/triplets.R
         , RLOG =C:/Users/userid/Documents/My SAS Files/R/submit_r.log
         , gpath=C:/Users/userid/Documents/My SAS Files/R
         )
```

The results of R execution are written to the file `C:/Users/userid/Documents/My SAS Files/R/submit_r.log` and repeated below.

```
/******************************** Begin R Output ******************************\
> setwd( "C:/Users/userid/AppData/Local/Temp/SAS Temporary Files/Tempdir" )
> library(scatterplot3d) ;
> triplets <- function( m, n )
+ # compute Pythagorean triplets
+ {
+ a2 <- ( 1 : m )^ 2 ;
+ b2 <- ( 1 : n )^ 2 ;
+ a2_b2 <- outer( a2, b2, "+" ) ;
+ c <- ( sqrt( a2_b2 ) %% 1) == 0 ;
+ indices <- matrix( 0, m*n, 3 ) ;
+ for( i in 1 : m )
+ for( j in 1 : n )
+ if( c[ i, j ] ) indices[ i, ] <- c( i, j, sqrt( a2_b2[ i, j ] )) ;
+ triplets <- indices[ indices[ , 1 ] > 0, ] ;
+ } ;
>
> # SAS macro vars supply values to R function via substitution ;
> # m = 50, n = 50 ;
> trio <- triplets( 50, 50 ) ;
> print( trio[ 1:5,] ) ;
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    4    3    5
[3,]    5   12   13
[4,]    6    8   10
[5,]    7   24   25
> write.csv(trio, file="trio.csv" ) ;
> png( filename="C:/Users/userid/Documents/My SAS Files/R/Triplets_2D.png",
 bg="transparent" ) ;
> plot( trio[ , 1 ], trio[ , 2 ], main="Pythagorean Triplets 2d", xlab="a", ylab="b",
 pch=20, type="p" ) ;
> s2d.coords<-xy.coords( trio[ , c( 1, 2 ) ] ) ;
> text( s2d.coords$x, s2d.coords$y, cex=1,  labels=as.character( trio[ , 3 ] ), pos=2 ) ;
> dev.off() ;
null device
          1
> png( filename="C:/Users/userid/Documents/My SAS Files/R/Triplets_3D.png",
 bg="transparent" ) ;
> s3d<-scatterplot3d( trio, xlab="a", ylab="b", zlab="c", main="Pythagorean Triplets 3D",
 pch=20, type="p" ) ;
> s3d.coords <- s3d$xyz.convert(trio) ;
> text( s3d.coords$x, s3d.coords$y, s3d.coords$z, cex=1,
 labels=as.character( trio[ , 3 ] ), pos=1 ) ;
> dev.off() ;
null device
          1
\******************************** End R Output *********h*********************/
```

The two- and three-dimensional scatterplots of the triplets are shown in Figures 2 and 3.

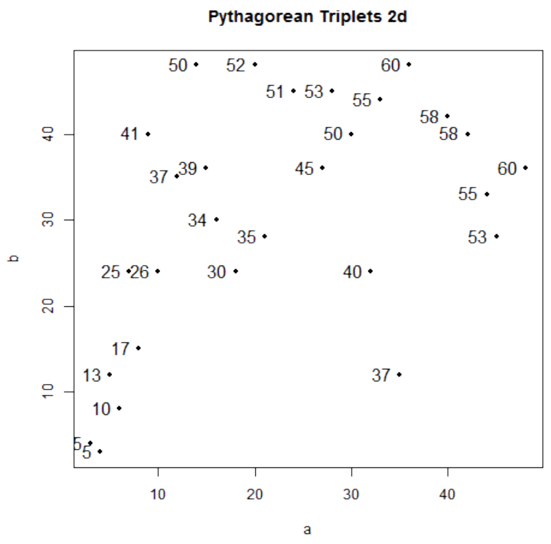

Figure 2: 2D Pythagorean Triplets

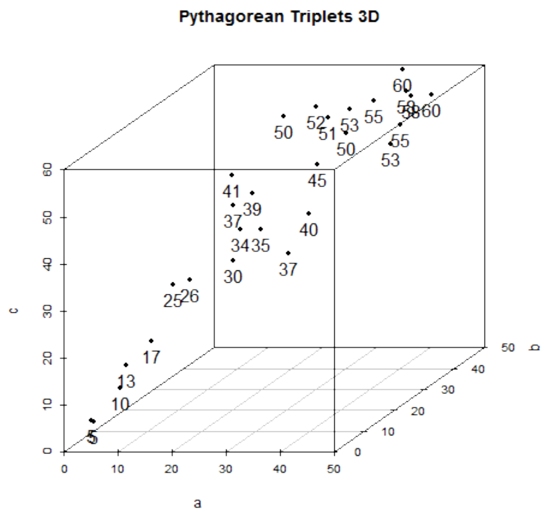

Figure 3: 3D Pythagorean Triplets

DESCRIPTION OF OPERATION

The `%SUBMIT_R` macro uses three auxiliary macros to assist its operation. They are briefly described below.

AUXILIARY MACROS

- The `%ExportDataSetToR` macro converts a SAS dataset into a CSV file using SAS' `PROC EXPORT` procedure. The CSV file is by default created in the SAS Temporary Files directory unless the macro parameter `&SAS_TEMP` is set to a user-specified directory path.
- The `%ImportDataSetFromR` macro converts an R data frame into a SAS dataset using SAS' `PROC IMPORT` procedure. The R data frame is converted to a CSV file in the R working directory and `PROC IMPORT` is used to convert the CSV file into a SAS dataset.
- The `%R` macro manages the creation of an R script. It produces a global macro variable called `&R_SCRIPT`. `&R_SCRIPT` contains R statements that are assembled into an R script and submitted to the R executive for

interpretation. The `%R` macro uses the semicolon (;) to delimit R statements, so each statement must be terminated by a semicolon when the `%R` macro is used.
Nota bene: The `%R` macro must be used *before* the `%SUBMIT_R` macro is invoked when creating an R script within a SAS program.

THE `%SUBMIT_R` MACRO

The `%SUBMIT_R` macro manages the tasks associated with executing the R script submitted to it. The script may be represented as a SAS macro variable or read from a user-specified file. It is transformed into an executable R script, which `%SUBMIT_R` passes to the R executive using the unnamed pipe mechanism [SAS(2013)], and manages the textual and graphic output produced as a result of execution.

1. Macro variables are initialized: the parameters `&RSCRIPT, &RLOG, &RCMD, &ROPTS, &SETWD, &GPATH` are checked, and if they are null, then they are assigned default values. Windows path names are translated: "\" is changed to "/" to comply with R convention.

2. Names for the R input and output files are created: the input file is `&SASTEMP/S_U_B_M_I_T_R.R` and the output file is `&SASTEMP/S_U_B_M_I_T_R.Rout`.

3. If the `&RSCRIPT` global macro variable is defined as a result of using the `%R` macro, the statements that it contains are written to the R input file. If the macro variable is not defined, it is assumed that R statements come from a user-specified file. In this case, statements from the user-defined file are written to the R input file.

4. The `&SETWD` parameter is checked for non-null contents. If it is empty, the SAS Temporary Files directory is used as the R working directory. If it is non-null, it represents the pathname of the R working directory. The R `setwd` command is written to a prologue file that will precede the R input file in providing statements for execution.

5. The R input file is scanned for the `%ExportDataSetToR` command. This command is case-insensitive, so that `exportdatasettor` and `ExportDataSetToR` and `EXPORTDATASETTOR` are all textually equivalent. If the command is detected in the input file, it is parsed and the SAS dataset and R data frame names are extracted from the text. A `read.csv` statement is assembled containing the names of the SAS dataset from which to read and the R data frame to be created. It is written to the prologue file that already contains the `&SETWD` statement. The SAS data step function `call execute` is used to position the `%ExportDataSetToR` macro for invocation after the data step performing the processing exits.

   The input file is similarly scanned for the `%ImportDataSetFromR` command. If it is found, it is parsed and the SAS dataset and R data frame names are extracted from the text and stored in a macro variable for later use.

   More than one `%ExportDataSetToR` and/or `%ImportDataSetFromR` command may be used in an R script input to `%SUBMIT_R`.

6. R statements are read from the prologue and input files, respectively, into a SAS dataset called `r_in`. If `%ExportDataSetToR` macro statements are detected, they are deleted because the macro will already have been invoked to create CSV files in the R working directory. If `%ImportDataSetFromR` macro statements are detected, they are replaced with equivalent `write.csv` statements to convert R data frames into CSV files for later importation as SAS datasets.

7. The SAS dataset `r_in` is written to the R input file, `S_U_B_M_I_T_R.R`.

8. The unnamed pipe statement [SAS(2013)] is used to spawn a process that will invoke the R executive to interpret and execute the statements contained in the `S_U_B_M_I_T_R.R` file.

9. The `%ImportDataSetFromR` statements that were stored in a macro variable are parsed within a SAS data step, and for each dataset/data frame pair, a `call execute` statement is assembled that will invoke the `%ImportDataSetFromR` macro after the data step exits.

10. The contents of the `S_U_B_M_I_T_R.Rout`, are written to the SAS log or to the user-specified file named in the `&RLOG` macro parameter.

11. The R working directory is searched for bitmap-formatted files. If you created a graphics image using the `bmp`, `jpeg`, `png`, or `tiff` functions, the default Windows file extensions are `.bmp`, `.jpg`, `.png`, and `.tif`, respectively. If at least one bitmap-formatted file is found, an `HTML` file is created using the SAS Output Delivery System and the image is displayed in the Results Viewer window.

SUMMARY

The `%SUBMIT_R` SAS macro is an efficient interface between SAS and the R environment. By using the `%SUBMIT_R` macro, you can pass data from SAS to R and back to SAS, or create data in the R environment and return R data frames to SAS as datasets for further use. The `%SUBMIT_R` macro lets you create plots and graphs in the R environment and then display them in the SAS Results Viewer.

Since R is open source and is free software, it represents an effective augmentation to SAS.

ACKNOWLEDGMENTS

We thank Joseph Naraguma, Anita Rocha, and Susan Slaughter for their thoughtful comments and suggestions.

The technique of using SAS' unnamed pipe statement is central to using `%SUBMIT_R` and is pivotal to its operation. It was used in Xin Wei's `%PROC_R` macro [Wei(2012)]. Methods in this paper also borrow from the idea of creating an `HTML` file that references graphic image files produced by R from Phillip Holland's prior work [Holland(2005)].

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author:	Name	Ross Bettinger
	Enterprise	Consultant
	City, State, ZIP	Silver Spring, MD 20910
	E-mail:	`mailto:rsbettinger@gmail.com`

REFERENCES

[Holland(2005)] Holland, Phillip R., (2005). "SAS to R to SAS". PhUSE Proceedings.
`http://www.lexjansen.com/phuse/2005/cc/cc03.pdf`

[SAS(2013)] SAS Institute, Inc., (2013). "SAS Help and Documentation". SAS Institute, Cary, NC, USA.
`http://support.sas.com/documentation/93/index.html`

[Weisstein(2014a)] Weisstein, Eric W., (2014). "Archimedean Spiral." From MathWorld–A Wolfram Web Resource.
`http://mathworld.wolfram.com/ArchimedeanSpiral.html`

[Weisstein(2014b)] Weisstein, Eric W., (2014). "Nested Radical." From MathWorld–A Wolfram Web Resource.
`http://mathworld.wolfram.com/NestedRadical.html`

[Wei(2012)] Wei, Xin, (2012). "`%PROC_R`: A SAS Macro That Enables Native R Programming in the Base SAS Environment", Journal of Statistical Software, January 2012, Volume 46, Code Snippet 2.
`http://www.jstatsoft.org/v46/c02`