Journal of Statistical Software

January 2012, Volume 46, Code Snippet 2. *http://www.jstatsoft.org/*

%PROC_R: A SAS Macro that Enables Native R Programming in the Base SAS Environment

Xin Wei
Roche Pharmaceuticals

Abstract

In this paper, we describe %PROC_R, a SAS macro that enables native R language to be embedded in and executed along with a SAS program in the base SAS environment under Windows OS. This macro executes a user-defined R code in batch mode by calling the unnamed pipe method within base SAS. The R textual and graphical output can be routed to the SAS output window and result viewer, respectively. Also, this macro automatically converts data between SAS datasets and R data frames such that the data and results from each statistical environment can be utilized by the other environment. The objective of this work is to leverage the strength of the R programming language within the SAS environment in a systematic manner. Moreover, this macro helps statistical programmers to learn a new statistical language while staying in a familiar environment.

Keywords: SAS, R, macro, pipe, ODS, batch mode, HTML.

1. Introduction

As the number of data analysis languages and platforms rapidly grows, it is becoming increasingly desirable for programmers to call a language from a different environment. Numerous efforts in this direction have empowered statistical analysts with the ability to use their favorite language within their operating environment. The R project, a leading open-source statistical computing language and environment (R Development Core Team 2011b), has packages to integrate native C++ (Eddelbuettel and Francois 2011) and SQL code (Grothendieck 2011). Python programmers have developed various methods that allow rich statistical collections of the R project to be accessible within the Python environment (Xia, McClelland, and Wang 2010). In the proprietary arena, SAS is a widely used system for data management and statistical analysis. PROC SQL is SAS's product to enable SQL programmers to efficiently perform data manipulation within the SAS platform without much base SAS knowledge (SAS Institute

Inc. 2004). In light of increasing momentum of R in statistical computing, the SAS institute recently released the interface between R and SAS/IML (Wicklin 2009), the latter being a specialized SAS module oriented towards matrix computation. However, base SAS, which is the foundation of all SAS modules and has a much bigger user community than SAS/IML, lacks a well documented and user-friendly integration for R programming. A programming interface of base SAS and R is highly desirable because base SAS and SAS/STAT users frequently need to access the novel statistical/visualization tools that have been developed in the open source R project and not yet been implemented in SAS. In this paper, we present a SAS programming interface in which R code can be written and executed within a SAS session and the resulting textual and graphical output of R is accessible from a SAS terminal. This SAS macro constructs a functional R script based upon a built-in R function and a customer-defined R script. Upon submission, the R code is executed via a batch mode session by a SAS pipe. From the pipe, textual and graphical R output can be rendered to the proper SAS terminal. We name this interface `%PROC_R` in the same spirit as the commercialized SAS products such as `PROC SQL`.

2. Interface

sqldf (Grothendieck 2011) is an R package from which SQLite syntax can be submitted in R code to manipulate R data frame. The following code demonstrates the `select` clause of SQL is used in R environment to compute summary statistics of R data frame `testdata`.

```
sqldf(" select avg ( value ) mean from testdata group by treatment")
```

Similarly, the syntax of `PROC SQL` from base SAS is as follows:

```
proc sql;
        Select mean ( value ) as mean from testdata group by treatment;
quit;
```

We design our SAS/R interface in the similar fashion to **sqldf** and `PROC SQL`. Following a regular SAS program, a R script can be written and embedded in the following approach:

```
%include "C:\Proc_R.sas";
%Proc_R (SAS2R =, R2SAS =);
Cards4;

*****************************
***Please Enter R Code Here***
*****************************

;;;;
%Quit;
```

Following a chunk of regular SAS code, `%include` statement is invoked to execute the source code of `%Proc_R` that prepares the proper environment in the current SAS session for R code construction and execution. Note that the macro `%PROC_R` has two macro variables `SAS2R`

and R2SAS as input parameters that define the data transfer mechanism between SAS and R. SAS2R specifies the names of SAS datasets to be used in R session after being converted to R data frame. R2SAS defines the R data frames resulted from the R session that needs to be converted to SAS dataset because they may be needed in the remaining SAS session for additional analysis. Next we use four examples to demonstrate how the workflow of SAS and R could be seamlessly glued together by this interface.

3. Examples

3.1. Compute eigenvalue and eigenvectors for a numeric SAS dataset

R is a matrix oriented language while the base SAS is designed more toward the handling of tabular dataset. In this example, we first use SAS to create a SAS dataset "test" which mimics a 10 X 10 numeric matrix. In the %PROC_R macro argument, we specify this test SAS dataset as the input data for R session. This SAS dataset is converted to an R data frame on the backend of %PROC_R. Since data frame is the default R object to which SAS dataset is converted, one needs to explicitly convert the data frame to a matrix object in the user-defined R script. Finally eigen() function of R is invoked to compute the eigenvalue and vectors.

```
data test;
  do x=1 to 4;
     array a[4] a1-a4;
         do i=1 to 4;
        a[i] = rannor(100);
         end;
         output;
  end;
         drop i x;
run;

%include "C:\Proc_R.sas";
%Proc_R (SAS2R = test, R2SAS =);
cards4;

R> testm <- as.matrix(test)
R> eigen(testm)

;;;;
%quit;
```

The R log and result page are displayed on the SAS output window(Figure 1).

3.2. Conduct regular statistical analysis with R from SAS

This example uses R example from Venables and Ripley (2002). As seen below, several lines of example R codes are executed by %PROC_R macro.

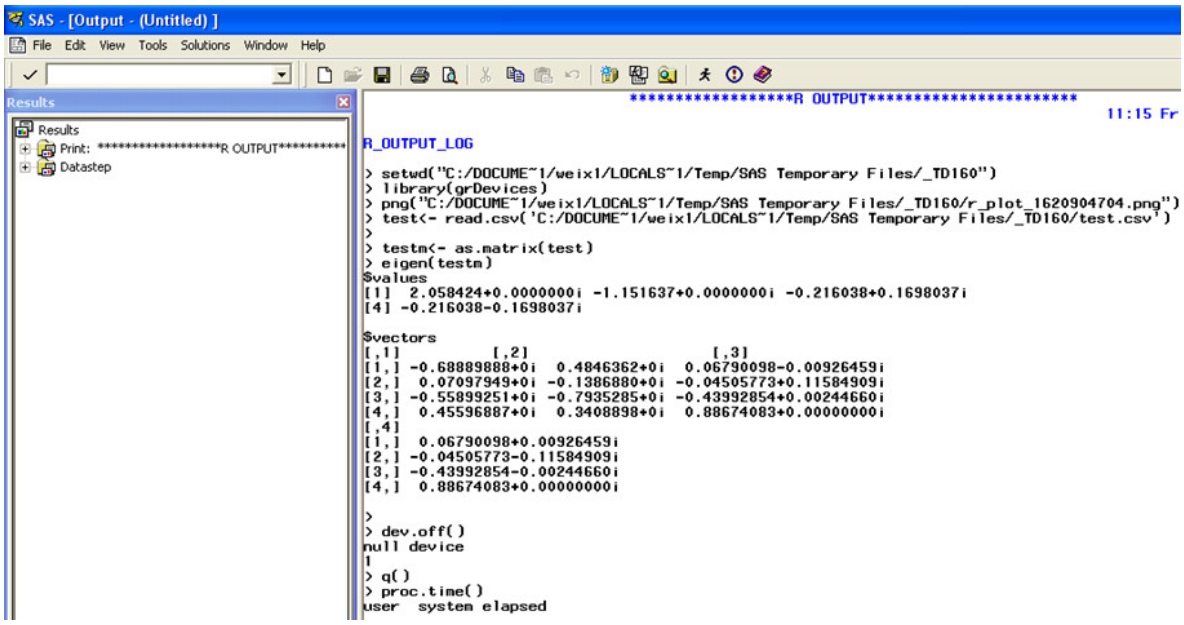

Figure 1: R matrix computation result is displayed in SAS output window.

```
%include "C:\Proc_R.sas";
%Proc_R (SAS2R =, R2SAS =) ;
cards4;

R> x <- seq(1, 25, 0.5)
R> w <- 1 + x / 2
R> y <- x + w * rnorm (x)
R> dum <- data.frame (x, y, w)
R> fm <- lm (y ~ x, data = dum)
R> summary (fm)
R> fm1 <- lm(y ~ x, data = dum, weight = 1 / w ^ 2)
R> summary (fm1)
R> lrf <- loess(y ~ x, dum)
R> plot (x, y)
R> lines(spline(x, fitted(lrf)), col = 2)
R> abline(0, 1, lty = 3, col = 3)
R> abline(fm, col = 4)
R> abline(fm1, lty = 4, col = 5)

;;;;
%quit;
```

R session on the backend creates the graphics for linear fit, loess smoothing and resistant regression that are displayed on either SAS graph editor or result viewer. The statistical outputs are printed on the SAS output window(Figure 2).

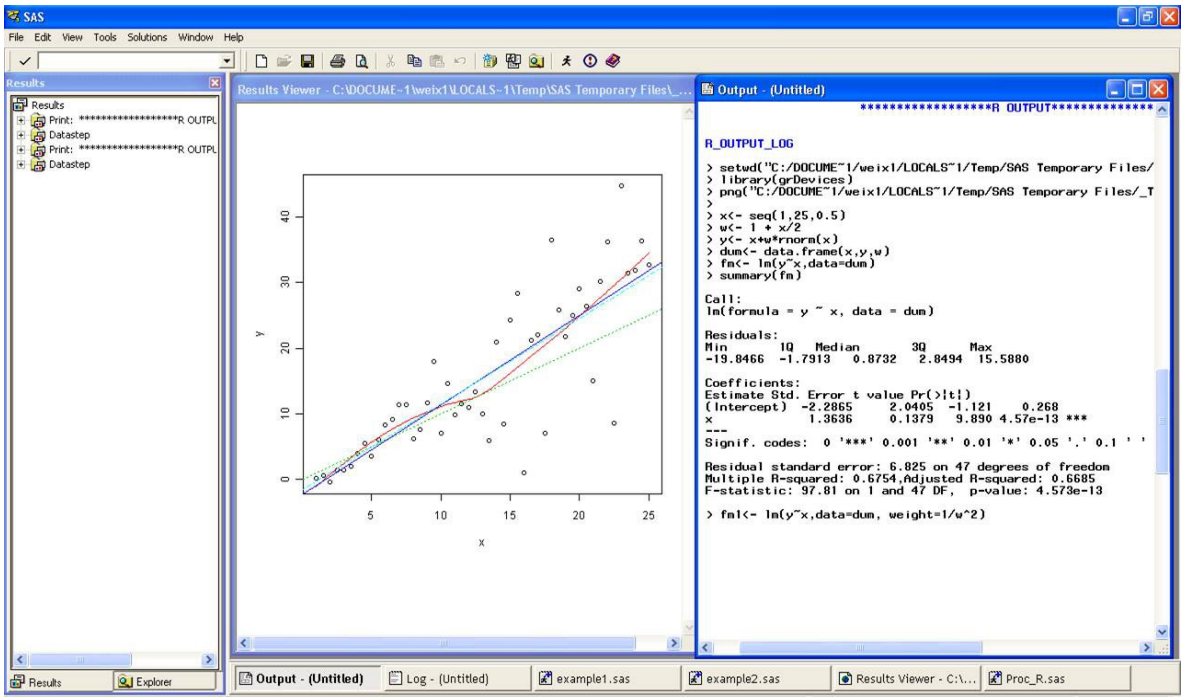

Figure 2: R statistical result and graphics are displayed in SAS platform.

3.3. Creation of R animation in base SAS environment

R wiki page provides a simple piece of code using the **caTools** package (Tuszynski 2011) that produces a flashy animation for Mandelbrot set (Tuszynski 2010). This code can be executed from `%PROC_R` interface without any modification and the resulting gif animation can be viewed from SAS result viewer (Figure 3).

```
%include "C:\Proc_R.sas";
%Proc_R(SAS2R=,R2SAS=);
cards4;

R> library("caTools")
R> jet.colors <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
  "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
R> m <- 1200
R> C <- complex(real = rep(seq(-1.8, 0.6, length.out = m), each = m),
  imag = rep(seq (-1.2, 1.2, length.out = m), m))
R> C <- matrix(C, m, m)
R> Z <- 0
R> X <- array(0, c (m, m, 20))
R> for (k in 1: 20) {
  Z <- Z^2 + C
  X [, ,k] <- exp(-abs(Z))
}
R> write.gif (X, "Mandelbrot.gif", col = jet.colors, delay = 100)
```

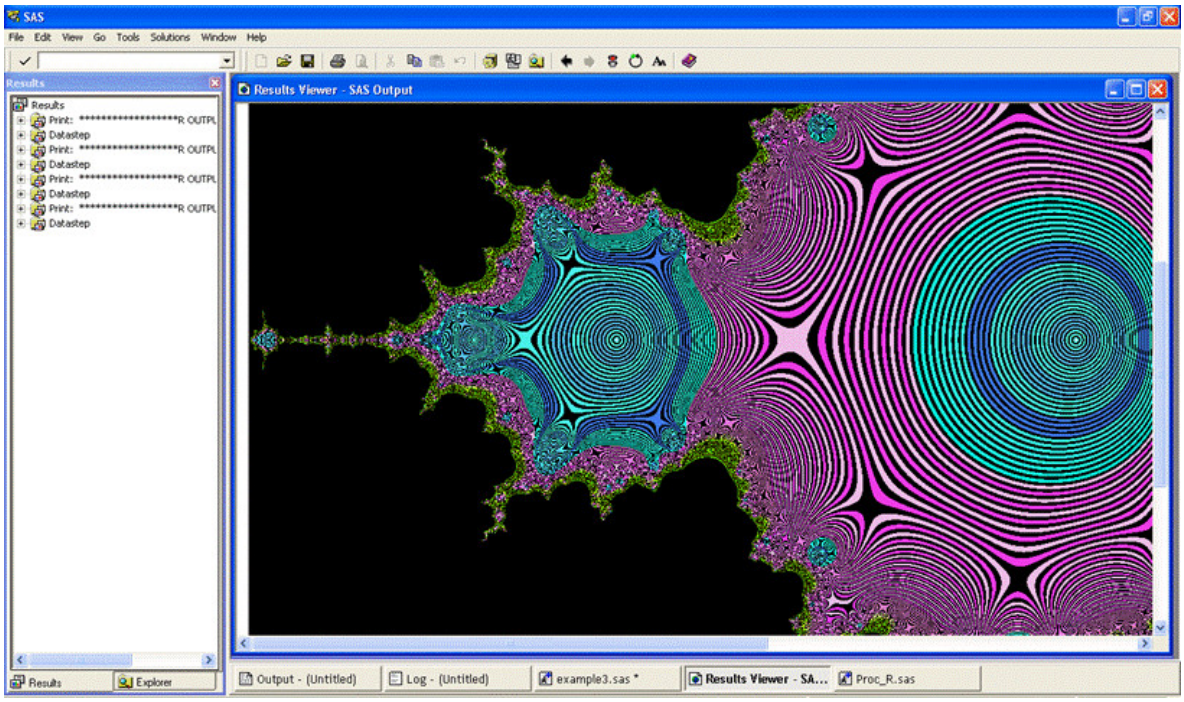

Figure 3: R gif animation is produced and displayed in SAS platform.

```
;;;;
%quit;
```

3.4. Programmatically integrate SAS and R output

In this example, I demonstrate that an R graphs is first produced by %PROC_R interface and then seamlessly written into an HTML report created by SAS ODS. The following ODS statement produce a sample HTML report:

```
ODS HTML FILE="c:\example.html" STYLE=minimal gpath= "c:\" GTITLE GFOOTNOTE;
proc print data=sashelp.company(obs=10); run;
ods _all_ close;
```

What follows is the R code that we copied from Ruckstuhl and Locher (2011), using packages **IDPmisc** (Locher, Ruckstuhl *et al.* 2011) and **SwissAir** (Locher 2011), and submitted in the %PROC_R interface.

```
%include "C:\Proc_R.sas";
%Proc_R(SAS2R=,R2SAS=);
cards4;

R> setwd("c:/RINSAS")
R> library("IDPmisc")
```

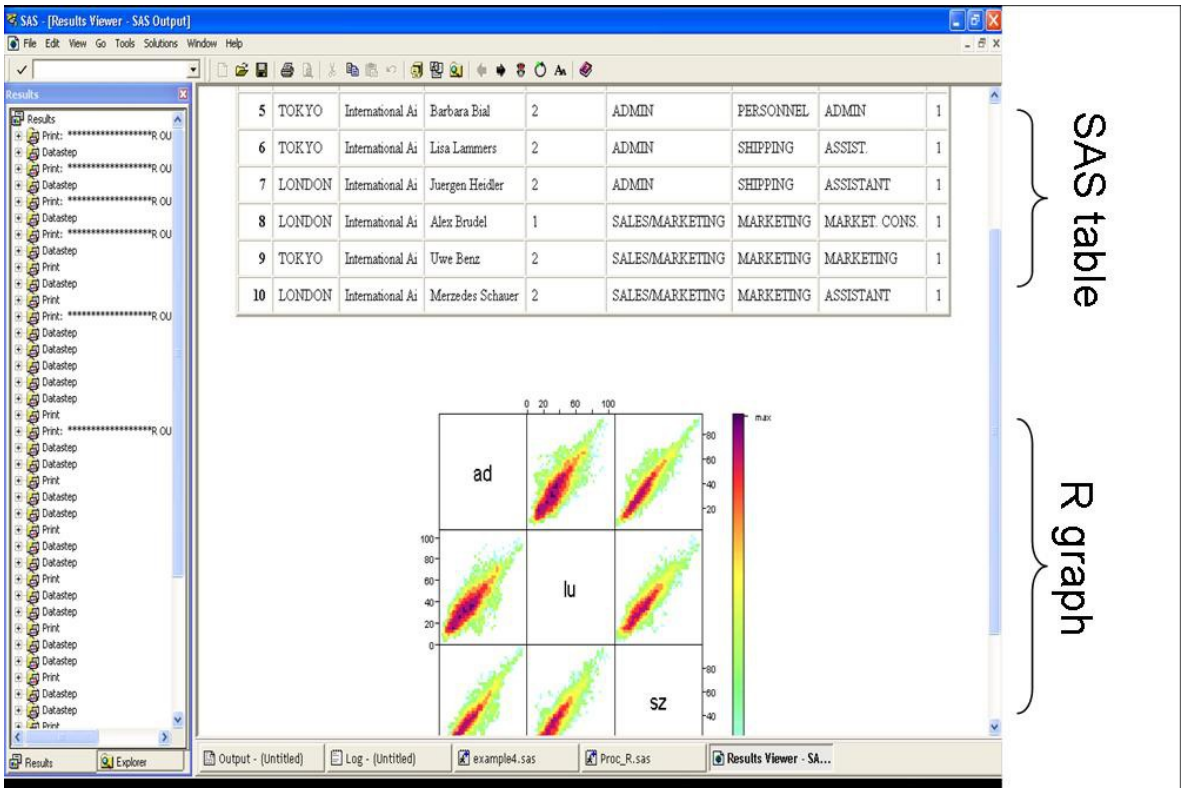

Figure 4: R graph is integrated in a SAS HTML report.

```
R> library("SwissAir")
R> Ox <- AirQual[, c ("ad.O3", "lu.O3", "sz.O3") ] +
  AirQual[, c ("ad.NOx", "lu.NOx", "sz.NOx") ] -
  AirQual[, c ("ad.NO", "lu.NO", "sz.NO") ]
R> names(Ox) <- c ("ad", "lu", "sz")
R> ipairs(Ox, ztransf = function(x) { x [ x < 1 ] <- 1; log2 (x) })

;;;;
%quit;
```

As we may see in the next sections, the resulting R scatter plot is saved to either the default or user defined location. The following SAS code post-processes the example SAS HTML file so that R graphics is integrated in it (Figure 4).

```
data html;
     infile "c:\example.html" truncover lrecl = 256;
     input raw $ 1-200;
run;
data html;
     set html;
     output;
     if raw = "<br>" then do;
```

```
            output;
            raw = '<div align="center">'; output;
        raw = "<img src=&inhtml border='0' class='c'>"; output;
     end;
run;
data _null_;
    set html;
    file "c:\example.html";
    put raw;
run;
```

4. Implementation

The %PROC_R program accomplishes R analysis in the base SAS via six steps:

- Write a draft R program based on user supplied R code and system built-in R function.
- The draft R script is interrogated and modified by SAS macro so that it can be executed in SAS environment
- The data exchange mechanism between SAS and R is implemented in both SAS code and R code.
- The completed R script is submitted to a backend R session via batch mode by base SAS pipes.
- The automatically produced text file containing R statistical output and log is transformed to SAS data and printed on the the base SAS output window.
- The R graphics is either explicitly or implicitly saved to a default or customer specified directory and routed to SAS graph window or result viewer via HTML format.

4.1. R code construction

This SAS program enables users to enter R script via `infile` and `cards4` statement in SAS data step. `%PROC_R` is a very simple macro that only specifies that R code is saved at the default SAS workspace and provide the first several lines of data step. `Cards4` (or `datalines4`) statement is used here because it allows semicolon as the part of input data when semicolon is recognized as the end of statement by both SAS and R. The four consecutive semicolons at the end mark the end of user define R scripts. While this manuscript is under development, we noticed that this technique has also been discussed in some techinical blogs (Xie 2011). The `run` statement of this data step is provided in the %Quit macro which is the main body of this interface and will be described in more details below.

4.2. R code refinement

In the `%Quit` macro, SAS program will inspect the resulting R program to determine what change needs to be made in order for the R program to be submitted from SAS environment.

First of all, the users are encouraged to define their own user space to store the log, graphics and intermediate datasets by `setwd()` function. If `setwd()` is not detected in user defined R code or is commented out, SAS program will add one line of `setwd()` function in the R code that set the R workspace to the current SAS temporary directory. Secondarily, if users do not explicitly write the graphics as a permanent file and instead intend to view them on fly, SAS program would add the lines to start R graphics device, save the figure and close the device. Later I will show how the saved R graphics can be viewed from SAS end.

4.3. Data exchange between R and SAS using CSV as intermediate format

We made conscious decision to make the macro arguments as simple as possible in order to allow less experienced SAS user to utilize this interface when they need to operate in SAS environment. The only arguments for this macro are to define the SAS and R datasets that are to be utilized by their counterpart. One or multiple SAS dataset names can be assigned to macro variable `SAS2R`. The presence of these SAS dataset names triggers a compilation of SAS do loop to sequentially write the listed SAS datasets to CSV files saved under user specified folder or SAS temporary directory. Then the proper R function like `read.csv` will be added to the current R codes to read these tab delimited files into R data frames with the same names. Therefore, users can directly call the corresponding R data frames using the SAS names defined in `SAS2R` argument without any change. Finally, if the value of macro variable `R2SAS` is not null, then its value, a list of names for R data frames, will be written into csv files by `write.csv` function which will be appended to the end of R codes. These csv files from R data frames are again read into SAS so that they can be used in the remaining the SAS program. Note that the naming convention for SAS datasets and variables need to be enforced throughout this process because not all R data or variable names are legitimate in SAS.

4.4. R submission in batch mode via pipe

Unnames pipe is a powerful SAS tool that execute a foreign program outside of SAS. Here we used it to execute the batch mode R scripts in parallel to the current SAS session (Wei 2009; R Development Core Team 2011a). As shown below, this pipe defines both the source R codes and export output/log file in ASCII format. `NOXWAIT` options force SAS session to suspend until the completion of the piped R process. I create a SAS macro variable `&rpath` to store the full path for R executable files. Currently, the paths for various R versions from 2.11.0 to 2.14.0 are provided so that users could easily adopt this macro to their own R version.

```
options noxwait xsync;
filename proc_r pipe "&rpath CMD BATCH
                     --vanilla --quiet
                     &source_r_code  &output_r_log";
data _null_;
     infile proc_r;
run;
```

4.5. Display of R log and graphics on SAS terminal

One primary motivation for SAS user to call R function from within SAS environment is

probably the state-of-art graphical capability of R. Therefore, it is priority of this interface to have a mechanism that channels the saved R graphics back to SAS session so that users can conveniently view the graphical output in real time. This can be achieved in two ways. First is to use the GSLIDE procedure to display the saved R figure on the SAS graph editor as shown in the following code:

```
goptions reset = all device = win iback = "&rdirec\&fgname"
         imagestyle = fit;
proc gslide; run; quit;
```

(Holland 2008) proposed another rather creative solution that is to create an HTML file with the link to the saved R graphics. The following SAS code creates an empty HTML, adds a link to the saved R graphics and render the HTML file to SAS result viewer. This means that one does not even need SAS/GRAPH installed to produce high quality graphics in SAS environment. We adopt this approach in our implementation. However, I do notice rare occasions where the graphics is not properly displayed in HTML page due to the excessively long string of file path and name particularly when the R results are stored in SAS temporary workspace. The remedy of this problem is to use `setwd` function in R script to save R result to a user defined destination with short path or simply SAS graph viewer instead of HTML reviewer to display R graphics.

```
ODS html file = "&sasdirec\rhtml_&nowtime..html" STYLE=minimal
    GPATH="&sasdirec\" GTITLE GFOOTNOTE;
ods listing close;
%global inhtml;
%let inhtml=%bquote("&sasdirec\&fgname");
DATA _NULL_;
FILE PRINT;
PUT "<IMG SRC=&inhtml BORDER='0'>";
run;
ods html close;
ods listing;
```

Similarly, since R log file has been saved under the R batch mode, it is straightforward to print this ASCII file on SAS output window after converting it to SAS dataset.

4.6. File management of SAS/R working space

The source R script, R log message and R graphical output are all saved under SAS temporary folder unless users defined otherwise. It is preferable to assign session specific names to these files so that the code and output from previous session would not be mistakenly called by the current session which may abort and produce no result due to syntax error. This can be achieved by appending a 12 digit number representing SAS system time to the end of file name. In one SAS session the users may explicitly write graphics to a permanent file or let SAS program do that on the backend. Thus the name of graphics could be either user defined or system assigned. In order to intelligently determine what the proper file is to visualize from SAS end, we again use SAS pipe function to scan the working space of the current

SAS/R session. The name of files in the directory is listed in a SAS dataset according to the descending order of the creation time, as demonstrated by the following code (Wei 2009).

```
FileName MyDir Pipe "dir &_sasdirec /a:-d";
data file;
  infile MyDir lrecl=300 truncover;
  input @1 file $100. @;
  format   file $100.;
  crtime = substr (file, 1, 20);
  if trim (left (scan (lowcase (file), 2, '.')))
     in ('gif', 'png', 'jpeg', 'jpg', 'ps')
  then do;
     _crtime = input (crtime, mdyampm.);
     temp = tranwrd (file, trim (left (crtime)), '*') ;
     temp = scan (temp, 1, '*') ;
     filename = trim (left (scan (temp, 2, ' ')));
  end;
run;
proc sort data=file; by descending _crtime descending filename;
      where trim (left (scan (lowcase (file), 2, '.')))
            in ('gif','png','jpeg','jpg','ps');
run;
```

Because the system time prior to and after the submission of the current R batch mode can be recorded and compared to the creation time of image file, it is easy to parse out the name of the newly created image in the last R session with the following SAS code. If new graphics is produced during the latest SAS/R session, then the macro varialbe `&fgsw` is set to one and SAS code from sction 4.5 will be conditionally run to display the figure in SAS graph window or HTML. Otherwise, if no new figures are detected then no further action is taken.

```
data _null_;
    call symput
        ('beforetm', trim (left (put (datetime (), datetime19.))));
run;
data _null_;
     set file;
         if _n_ = 1 then do;
        call symput ('fgsw', put (input ("&beforetm", datetime19.)
                     < (input (crtime, mdyampm.) + 60), best.));
        temp = tranwrd (file, trim (left (crtime)), '*');
        temp2 = scan (temp, 1, '*');
        fgname = scan (temp2, 2, ' ');
        call symput ('fgname', trim (left (fgname)));
         end;
run;
%put &fgname;
```

5. Discussion

Nowadays data analyst very often needs access to various tools/languages to accomplish complex task. Perl and Python is widely used to pre-process dataset prior to the advanced statistical analysis and visualization in R. Many SAS users are also very keen to learn R language in order to test novel statistical methodology developed in open source R that is outside the horizon of SAS/STAT. It is not rare for a multilingual data analyst to perform some analysis in one environment, save the intermediate datasets and continue the rest of analysis in a different environment. However, it would be difficult to keep track of analysis workflow if multiple computing environments are involved. Switching platform also adds to difficulties for others to repeat one's research. In this work, we developed the SAS macro that allows R analysis to proceed along with SAS program without moving out of SAS environment. We also show the way to integrate R analysis result into a SAS statistical report programmatically. There are several issues worth of note. First of all, it is not possible to embed this interface into other SAS macro because `cards` or `datalines` statement for R code input is not allowed within a SAS macro. We also believe that it is a good practice to keep SAS and R code block clearly separate even when they are in one program. Secondly, the whole chunk of R scripts written by users will be executed from start to end in one R batch mode. It is not feasible to test the R code piece by piece in this interface because the intermediate results would be lost at the end of each batch mode. Therefore this interface is suitable to test or run a small task or execute a large project that has been well developed in a more suitable R interactive development environment.

Acknowledgments

I appreciate Dr. Luis Tari's help with using LaTeX and Dr. Mick Kappler for his review of manuscript.

References

Eddelbuettel D, Francois R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. URL http://www.jstatsoft.org/v40/i08/.

Grothendieck G (2011). ***sqldf**: Perform SQL Selects on R Data Frames*. R package version 0.4-6.1, URL http://CRAN.R-project.org/package=sqldf.

Holland P (2008). www.hollandnumerics.co.uk/pdf/SAS2R2SAS_presentation.pdf.

Locher R (2011). ***SwissAir**: Air Quality Data of Switzerland for One Year in 30min Resolution*. R package version 1.1.01, URL http://CRAN.R-project.org/package=SwissAir.

Locher R, Ruckstuhl A, *et al.* (2011). ***IDPmisc**: Utilities of Institute of Data Analyses and Process Design*. R package version 1.1.16, URL http://CRAN.R-project.org/package=IDPmisc.

R Development Core Team (2011a). *An Introduction to R*. Vienna, Austria. ISBN 3-900051-12-7, URL http://www.R-project.org/.

R Development Core Team (2011b). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Ruckstuhl A, Locher R (2011). "Image Scatter Plot Matrix." http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=159.

SAS Institute Inc (2004). *SAS 9.1 SQL Procedure User's Guide*. Cary, NC. URL http://www.sas.com/.

Tuszynski J (2010). http://en.wikipedia.org/wiki/R_(programming_language).

Tuszynski J (2011). ***caTools**: Tools, Moving Window Statistics, GIF, Base64, ROC, AUC, etc.* R package version 1.12, URL http://CRAN.R-project.org/package=caTools.

Venables W, Ripley B (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York.

Wei X (2009). "Dynamically Allocating Exported Datasets by the Combination of Pipes and X statement." In *SAS Global Forum 2009 Proceedings*.

Wicklin R (2009). "Rediscovering SAS/IML Software: Modern Data Analysis for the Practicing Statistician." In *SAS Global Forum 2009 Proceedings*.

Xia XQ, McClelland M, Wang Y (2010). "**PypeR**, A Python Package for Using R in Python." *Journal of Statistical Software, Code Snippets*, **35**(2), 1–8. URL http://www.jstatsoft.org/v35/c02/.

Xie L (2011). www.sas-programming.com/2010/04/conduct-r-analysis-within-sas.html.

Affiliation:

Xin Wei
Pharmaceutical Research and Early Development Informatics
Roche Pharmaceuticals
340 Kingsland Street
Nutley, NJ, United States of America
E-mail: xin.wei@roche.com, xinwei@stat.psu.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 46, Code Snippet 2 *Submitted:* 2011-05-17
January 2012 *Accepted:* 2011-12-31