PhUSE 2011

Paper CS07

Calling R Functions from SAS

Dr. Peter Bewerunge, HMS Analytical Software, Heidelberg, Germany

ABSTRACT

This paper gives insights into the SAS interface to R which is available since SAS 9.22. The interface is realized as a SAS procedure called PROC IML. Here it is demonstrated how R Code can be submitted from SAS and how SAS data sets are send to R, used for computing and send back to SAS. Questions like “do SAS formats get lost on the way to R” and “how can R graphics be integrated into SAS PDF reports” will be discussed. Also best practice methods for error handling will be introduced.

WHAT IS SAS/IML?

The SAS **I**nteractive **M**atrix **L**anguage (IML) is a programming language for explorative data analysis. It implements a wide range of functions for most standard matrix operations. Also linear algebraic and statistical functions like solving ordinary differential equations or linear systems are part of SAS/IML. Direct access to SAS data sets e.g. editing or creating new ones is given. In addition own SAS/IML modules can be developed and integrated into applications. In addition SAS IML implements the interface to R which can be accessed from SAS via PROC IML.

This article focuses on the SAS procedure PROC IML and its features to integrate the R language. Special features of SAS/IML as well as the SAS/IML Studio which provides a dynamic and interactive interface to SAS/IML is not taken into account.

WHAT IS R?

The R environment with its R language is a freely available open source software tool which focuses on statistical computing and graphics. The syntax as well as the range of statistical functions is comparable to SAS/IML. One characteristics of R is its huge community of developers and users organizing themselves in different special areas, e.g. biologist and bioinformatics provide state of the art R algorithms at Bioconductor.org

WHY CALL R FUNCTION FROM SAS?

The question is why should one use PROC IML to call R functions from SAS when R is similar to IML? The probably most convincing argument is the really huge amount of free available state of the art algorithms as add-on packages for R. In particular R is very popular in research departments and is often used for the implementation of new algorithms or sophisticated plots. Also most journals in the area of software development for life science require the algorithms to be published and downloadable for everybody for reuse (e.g. Instructions to Authors, Bioinformatics: “Software or data must be freely available to non-commercial users”[1]).

Another advantage arises for users who switch from R to SAS. Here the R code has not to be converted into SAS but rather can be easily reused within SAS/IML.

REQUIREMENTS

INSTALLATION

SAS 9.22 which is the maintenance release 3 for SAS 9.2 is required. SAS/IML itself runs in 32-bit and 64 bit Windwos machines but the interface to R supports only 32-bit version of R.
The SAS workspace server and R must be installed on the same computer. It is also possible to access the SAS workspace server via Enterprise Guide®. In this case R must be installed on the SAS server.

R does not come along with the SAS installation and must be additionally downloaded (http://cran.r-project.org/). According to a SAS note (http://support.sas.com/kb/42/079.html) SAS/IML 9.22 does not support R versions starting with 2.12. The Version R-2.11.1 is recommended.

[1] http://www.oxfordjournals.org/our_journals/bioinformatics/for_authors/general.html

STARTING SAS WITH RLANG OPTION

By default SAS forbids to call R functionalities. This is specified by the systems option NORLANG. To get access to R, SAS has to be started with the RLANG Option (C:\...\sas.exe -RLANG).
Once SAS is started, one should prove whether R is installed and if access is given by the SAS system. This can be done by a PROC OPTION Step

```
PROC OPTIONS OPTION=RLANG;
RUN;
```

One of four possible messages is then printed to the SAS log and gives information about the access to R.

1. NORLANG: Do not support access to R language interfaces
2. ERROR: The RLANG system option must be specified in the SAS configuration file or on the SAS invocation command line to enable the submission of R language statements.
3. WARNING: SAS option RLANG is not supported on this host.
4. RLANG: Support access to R language interfaces

HOW TO SUBMIT R-CODE?

The PROC IML interface to R is very comfortable. R code can easily be submitted by the SUBMIT statement followed by the R option (SUBMIT / R). The end of the R code is completed by an ENDSUBMIT statement. Here a simple multiplication of two variables is shown:

```
PROC IML;
   SUBMIT / R;
      a  <- 2
      paste("a:",a)
      b  <- 3
      paste("b:",b)
      c <- a * b
      paste("c:",c)
   ENDSUBMIT;
QUIT;
```

The R output is directly printed to the SAS output:

```
[1] "a:  2"
[1] "b:  3"
[1] "b:  3"
```

HOW TO SUBMIT DATA FROM SAS TO R AND BACK?

One big challenge in developing interfaces is to match data from one format into another. Here SAS presents a user-friendly solution to transform data from the SAS world into the R world and back. This is realized by so called built-in subroutines which are implemented within IML.

DATA FROM SAS TO R

In the case of transferring data from SAS to R, this means to convert a SAS data set into an R data frame. A data frame in R is probably the most used object type in R. It is a matrix like collection of variables which can be from different data types.

In the following example the IML subroutine *ExportDataSetToR* transfers the SAS dataset SASHELP.Class into a R data frame named RDataFrame. The subroutine can be executed by the IML control statement “RUN.”

```
PROC IML;
   RUN ExportDataSetToR("SASHELP.Class", "RDataFrame" );
      SUBMIT / R;
         RDataFrame[1:4,1:5]
      ENDSUBMIT;
QUIT;
```

The R data frame output of rows 1-4 and columns 1-5 is then printed to the SAS output:

```
      Name Sex Age Height Weight
1   Alfred   M  14   69.0  112.5
2    Alice   F  13   56.5   84.0
3 Barbara   F  13   65.3   98.0
4    Carol   F  14   62.8  102.5
```

It should be mentioned that the same export of SAS/IML matrices into R matrices can be done by the subroutine *ExportMatrixToR*. Not demonstrated in this article.

DATA FROM R TO SAS

Also the other way around transferring data from R to SAS is implemented in an IML subroutine. Using the subroutine *ImportDataSetFromR* writes an R data frame to a specified SAS library.

In the following example only flowers from the type "setosa" are saved in a new data frame called "iris.setosa". Afterwards the new data frame will be written to the SAS library WORK. At the end PROC MEANS is used to prove that the data is really in the WORK library. For completeness it should be mentioned that the data frame "iris" is as popular for demonstration purpose in R as the SASHELP.class data set in SAS. It gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris.

```
PROC IML;
 SUBMIT / R;
    iris.setosa = iris[(iris$Species=="setosa"),]
 ENDSUBMIT;
RUN ImportDataSetFromR("WORK.IrisSetosa"
                     , "iris.setosa");
QUIT;

PROC MEANS data = WORK.IrisSetosa N SUM MEAN;
RUN;
```

A look into the PROC MEANS output confirms that the iris.setosa data frame generated within R was successfully saved to the SAS library WORK.

The MEANS Procedure

Variable	N	Sum	Mean
Sepal_Length	50	250.3000000	5.0060000
Sepal_Width	50	171.4000000	3.4280000
Petal_Length	50	73.1000000	1.4620000
Petal_Width	50	12.3000000	0.2460000

It should be mentioned that the same import for R matrices into SAS/IML matrices can be done by the subroutine *ImportMatrixFromR*. Not demonstrated in this article.

DATE, TIME, AND DATETIME VALUES

SAS and R have different data types. This comes especially true when working with SAS dates and times formats. R has nothing comparable to SAS Formats but implements classes to represent dates and times.

The conversion of this formats is like this:

- SAS date formats (e.g. DATEw.d) are converted to the R class "Date".
- SAS datetime formats (e.g. DATETIMEw.d) and SAS time formats (e.g. TIMEw.d) are converted to the R class that inherit from "POSIXt".
- In all other cases the SAS formats are assigned to the R class "numeric".

The conversion from R to SAS is like this:

- Variables form the class "Date" are converted to the SAS DATE9. format.
- Variables form the class "POSIXt" are converted to the SAS TIME19. format.
- In all other cases no SAS format is assigned.

GRAPHICS FROM R TO SAS

In principle, R graphics can be created during a PROC IML step. However this works only if you run SAS on your local machine. In this case also the local R is used to create a graphic, which is then displayed in a second R window on the same computer.

If one would like to view a R graphic in a second window you are bounded to your local SAS Display Manager. There´s no way to display a R graphic when connected to a SAS server via a client like the SAS Enterprise Guide, because in this case the SAS Server and also R is installed on a different computer than the client.

Thinking of a SAS program with different steps and e.g. R calls producing five different graphics via PROC IML makes clear, that just displaying the R graphics in a separated R window is not satisfying enough. We would like to have a physical copy of the graphic which could be integrated into e.g. a pdf-report. Therefore, the R graphic has to be streamed to a physical directory. This can be easily done via the R function png().

Here an example is given how to generate a graphic in R via PROC IML. The png()-function opens an output device for the graphic. This is followed by the code to produce the graphic. At the end the graphic device is closed by the dev.off() function. The original R code to produce the graphic is from the R graphic gallery[2]:

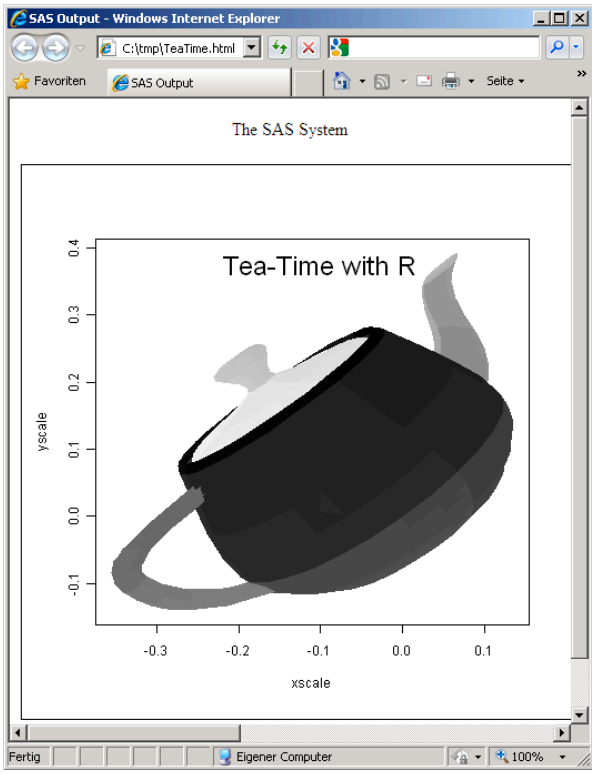
```
PROC IML;
      SUBMIT / R;
         png(file="c:/tmp/TeaTime.png", bg="transparent")
         library(ncdf)
         teapot<-open.ncdf("U:/Konferenzen/PhUSE/2011/Code/Teapot.nc")
         # file at http://www.maplepark.com/~drf5n/extras/teapot.nc
         edges<-get.var.ncdf(teapot,"tris")
         vertices<-get.var.ncdf(teapot,"locations")
         plotMesh<-function(vertices,edges,col,rot.mat=diag(4),dist=0.1,...){
            vertices<-ltransform3dto3d(vertices,rot.mat,dist)
            xscale <- range(vertices[1,])
            yscale <- range(vertices[2,])
            plot(xscale, yscale, type = "n")
            ord<-order(apply(edges,2,function(x){sum(vertices[3,x])}))
            if (length(col) == 1){
               sapply(ord,function(x){
               polygon(vertices[1,edges[,x]],vertices[2,edges[,x]],col=col,...)})
            } else {
               sapply(ord,function(x){
               polygon(vertices[1,edges[,x]],vertices[2,edges[,x]],col=col[x],...)})
            }
            invisible(ord)
         }
         grey.colors<-function(n,start=0,end=1){
            if(length(n)>1) n=length(n)
            gray(seq(start,end,len=n))
         }
         library(lattice)
         rot.mat <- ltransform3dMatrix(list(z=45,y=30)) #;rot.mat
         plotMesh(vertices,edges,rot.mat=rot.mat,col=grey.colors(dim(edges)[2]),lty=0)
         text(-0.1, 0.375, "Tea-Time with R", cex = 2)
         dev.off()
      ENDSUBMIT;
QUIT;
```

At the end, the resulting teapot graphic is embedded into a HTML ODS destination within a data null step:

```
ODS ESCAPECHAR='^';
ODS HTML FILE='c:\tmp\TeaTime.html' STYLE=minimal
GPATH='c:\temp\' GTITLE GFOOTNOTE;
```

[2] http://addictedtor.free.fr/graphiques/sources/source_134.R

```
DATA _NULL_;
   FILE PRINT;
   PUT "<IMG SRC='c:\tmp\TeaTime.png' BORDER='0'>";
RUN;
ODS _ALL_ CLOSE;
```



The teapot graphic was produced by R which was executed within a PROC IML step. Then the graphic was embedded into a HTML-report by SAS.

ERROR HANDLING

Error Handling is the mechanism to handle the occurrence of exceptions which interrupt the normal flow of a program. With SAS and R two different kinds of error handling mechanism have to be combined.

The interface PROC IML puts an error-message to the SAS-LOG if an error occurs during an R call. Besides looking to the SAS LOG, there are two obvious ways to notice in SAS that an exception has occurred during the execution of R Code. The first one is to read out the value of the global SAS macro variable *syserrortext*. The second option is to use the tryCatch-functionalities in R to catch up the error-message and send it back to SAS as a data set.

GLOBAL SAS MACRO VARIABLE SYSERRORTEXT

SAS recognizes in the global macro variable *syserrortext* the last error message that occurred in SAS. In the case of something has gone wrong during a PROC IML step *syserrortext* can be read out and the error can be handled. The example shows a not allowed division by an alphanumeric variable (1/"5") in R. If the error in R is thrown, the processing in R will be stopped and the print-function will not be executed anymore.

```
PROC IML;
   SUBMIT / R;
      1/"5"
      print("Here I am in R.")
   ENDSUBMIT;
QUIT;
```

Back in SAS the error is handled by putting the error message to the SAS LOG:

```
DATA _NULL_;
   IF ("&SYSERRORTEXT." NE "") THEN DO;
      PUT "Error in R: &SYSERRORTEXT.";
   END;
RUN;
```

The resulting message in the SAS LOG looks like this:

```
Error in R:  R: non-numeric argument to binary operato
```

For any reason, the last letter of the message did get lost on its way from R to SAS. SAS-LOG:

Be aware of the global macro variable *syserrortext* really saves the LAST error message that occurs during the processing in SAS. To be sure that the error really occurred during the PROC IML-Step it must be proven that this macro variable is really empty before.

TRYCATCH IN R

R comes along with a try-catch-mechanism to handle errors. Again the example shows a not allowed division by an alphanumeric variable (1/"5") in R, but now the error is handled by an error-handling-block. The error-message in R is saved by the variable *errMess*. This variable is then saved as a *data.frame* which at the end allows the use of the IML subroutine *ImportDataSetFromR* in order to save the message in a SAS data set.

```
PROC IML;
   SUBMIT / R;
      errMess = ""
      tryCatch({
               x = 1/"5";
            },
            error = function(ex) {
              errMess = geterrmessage()
              errMess  <<- as.data.frame(errMess)
            },
            finally = {
             cat("The error was handled \n")
            }
      )
   ENDSUBMIT;
   RUN ImportDataSetFromR("WORK.ErrMessFromR", "errMess");
QUIT;
```

Back in SAS, the error message is available in a SAS data set which can be used for further processing:

```
DATA _NULL_;
   SET WORK.ErrMessFromR END=LAST;
   IF LAST THEN DO;
      CALL SYMPUT('errMessInSAS', errMess);
   END;
RUN;
%PUT Error-Message from R: &errMessInSAS.;
```

SAS-LOG:

```
Error-Message from R: non-numeric argument to binary operator
```

Besides the effect that the lost last letter of the error message is present again, this kind of error handling allows a much better control of the program flow.

LIMITATIONS

Limitations of the interface PROC IML to R are:

- Only available for Windows computers.
- In the case of having different R installations on your computer, there´s no option to tell SAS which R to use.
- The use of large data sets is limited by the memory size of your computer. In R the data are loaded and processed in the main memory. The same is true for SAS IML.
- The R software comes along with absolutely no warranty. This should be taken into account especially for the large amount of add-on-packages.
- R provides a wide range of data types, because everything is an object in R. Every data type has to be converted either to a *data.frame* or a *matrix* before transferring data to SAS.

CONCLUSION

In conclusion SAS presents with PROC IML a very powerfull interface to the open source software R. Now SAS users have access to the state of the art algorithms which are available in the huge amount of R add-on packages.

The SUBMIT and ENDSUBMIT-Statement in the PROC IML step allows to execute R code. Also SAS data sets are easily send to R and back via the so called built-in subroutines *ExportDataSetToR* and *ImportDataSetToR* which is implemented within IML. The interface converts special SAS formats like the SAS date and datetime to the R counterpart.

A special feature of R is its graphic facility. Here it was shown, how R graphics can be exported and integrated into a SAS PDF report. Furthermore, it was demonstrated, how the error handling techniques from two different worlds can be used to handle errors in SAS coming from R.

CONTACT INFORMATION

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

Dr. Peter Bewerunge
HMS Analytical Software
Rohrbacher Str. 26
69115 Heidelberg
Work Phone: +49 6221 6051-158
Fax:+49 6221 6051-99
Email: peter.bewerunge@analytical-software.de
Web: www.analytical-software.de