Paper CC-106

Integrating Data and Variables between SAS® and R via PROC IML: Enable Powerful Functionalities for Coding in SAS

Feng Liu, University of North Carolina at Chapel Hill, NC
Ruiwen Zhang, Cary, NC

ABSTRACT

Programming in R provides additional features and functions which augment SAS procedures for many statisticians or scientists in areas like bioinformatics, finance, education, etc. It is in high demand for SAS users to be able to call R within their SAS programs. Though existing papers show how to import R data set into SAS or vice versa, there is a lack of a comprehensive solution to transfer variables from anything other than a data set. In this paper, we present solutions to use PROC IML to interface with R software which enables transferring variables and data sets transparently between SAS and R. We also provide examples of calling R functions within IML modules which offers much flexibility for coding in SAS, especially for big projects involving intensive coding. This can also be used to pass parameters from SAS to R. In this paper, you will see a step-by-step demonstration of a SAS project which integrates calling R functions via PROC IML.

INTRODUCTION

SAS/IML Studio first introduced a mechanism for calling R functions back in 2009. As of SAS 9.22, this feature is available in PROC IML, which offers convenient access for SAS users to utilize a large number of build-in functions as well as the user-contributed packages in R. SAS Interactive Matrix Language (IML) is an interactive programming language which can execute a command as soon as you enter and return results immediately. PROC IML has a similar dynamic programming environment as R and operates matrices and allows dynamic allocation of storages for matrices and modules automatically. On the other hand, you can read, create and update SAS data sets in PROC IML without ever using DATA step. With PROC IML as an interface, you can easily integrate your advanced R programs with SAS programs. In this paper, you will see how to exchange data sets and variables between SAS and R which is the most essential step to make transition between the two programming tools seamlessly. Some practical suggestions are also provided in the paper for achieving better performance when running on moderate to big size of data.

TRANSFER DATA SETS BETWEEN SAS AND R

You will see two convenient ways to transfer data sets between SAS and R. The first one is utilizing the common "work" directory to store the data sets that need to be transferred. In this example, we save the SAS data set as *csv* file, and then read the *csv* file later into R using *read.csv* function. This method can be generalized to transfer files other than *csv* file. As we can see, if we have the *csv* file, it can be put directly into the work library and so save the hassle for converting to SAS data format. Here is the example code.

```
%let dir = %sysfunc(getoption(work));     /*achieve the work directory path*/

proc EXPORT data=sashelp.cars
     outfile= "&dir\rindata.csv"
     DBMS=CSV REPLACE;

proc IML;
     submit /R;
          dataset = read.csv("rindata.csv")
          print(dataset)
     endsubmit;
quit;
```

The next step then transfers the R data back into SAS. R package “foreign” has the function to write R table into files and generate SAS code at the same time, so that the target SAS data set will be written when the generated SAS code has been run. The complete example is shown below. There is one thing we need to notice while we are coding R in SAS PROC IML. Though variable names are not case-sensitive in SAS, we need to match the cases of variable names explicitly as shown in SAS table when writing R code because R is case-sensitive.

```
%let dir = %sysfunc(getoption(work));

proc EXPORT data=sashelp.cars
    outfile= "&dir\rindata.csv"
    DBMS=CSV REPLACE;
run;

proc IML;
    submit /R;
        dataset = read.csv("rindata.csv")
        lm.MPG = lm(MPG_City ~ .,data=dataset)          /*call lm function in R*/
        yhat = fitted.values(lm.MPG)

        library(foreign)
        outdata = data.frame(pred = yhat, resid = r)
        write.foreign(df=outdata,datafile="routdata_out_data.csv",codefile="routdata_out_data.sa s",
        package="SAS")
    endsubmit;
quit;


data export_train ;
    INFILE "&dir\routdata_out_data.csv" DSD LRECL= 200;
    INPUT pred resid;
run;
```

Another intuitive way to transfer SAS data set to R data frame is to call the EXPORTDATASETTOR or EXPORTMATRIXTOR subroutine in PROC IML.

```
proc IML;
    run ExportDataSetToR("sashelp.cars", "indata");
    submit /R;
        print(indata)
    endsubmit;
quit;
```

There are also some genuine PROC IML subroutines available for the reverse way. ImportDataSetFromR and ImportMatrixFromR correspond to EXPORTDATASETTOR or EXPORTMATRIXTOR, and we don’t give the example codes here since the usage of the two subroutines is quite straightforward.

According to our experiences in practice, the first method is preferable while we need to transfer large data sets, especially when the data set is wide with many columns of variables. Because calling those genuine subroutines in PROC IML for large data can be slow.

TRANSFER VARIABLES BETWEEN SAS AND R

PROC IML offers simple interface for transferring macro variables between SAS and R, which greatly improves the readability and efficiency of coding across the two platforms.

Let’s see the following example which lists the specific SAS variables passed to R. “*_family”* and “*_inputs”* are the string variables defined in IML, and you can then use a SAS macro variable reference to retrieved them in R once they are specified in the “*sumbit /R”* statement. The example code demonstrates explicitly the way to transfer macro

variables from SAS to R. Please note that "*_inputs*" is a list of variable names from the data set, and we can easily get subsets of the original data set through specifying the variables in this way.

```
proc iml;
    /* Transfer SAS dataset into R */
    run ExportDataSetToR("sashelp.cars", "indata");       /* Transfer parameters into R */
    _family = "gaussian";
    _inputs = '"Make","Horsepower","Cylinders","MPG_City"';

    submit _family _inputs /R;
        varnames <- c(&_inputs)
        lm.MPG = glm(MPG_City ~., data=indata[varnames], family=&_family)
        yhat = fitted.values(lm.MPG)
        print(yhat)
    endsubmit;
quit;
```

Alternatively, you can import all parameters in IML to R workspace, and just need the "*" option in the "*submit /R*" statement as shown blow. In the following example, you can also see how to pass any SAS macro variables generally via PROC IML and then to R.

```
%let family = gaussian;
%let inputs =  'Make', 'Horsepower', 'Cylinders', 'MPG_City';

proc iml;
    /* Transfer SAS dataset into R */
    run ExportDataSetToR("sashelp.cars", "indata");
    /* Transfer parameters into R */
    _family  =  "&family";
    _inputs  =  "&inputs";
    submit * /R;
        varnames <- c(&_inputs)
        lm.MPG = glm(MPG_City ~., data=indata[varnames], family=&_family)
        yhat = fitted(lm.MPG)
        print(yhat)
    endsubmit;
quit;
```

CREATE SAS/IML MODULES THAT CALL R

We are now ready to move forward to create IML modules that can pass data and arguments into R and then obtain the results from R. This will be useful when we need to call some R routines recursively. The module in IML is defined using *START* and *FINISH* statements. Please see codes below.

```
proc iml;
    start GLMInR( dataset, inputs, family );
        run ExportDatasetToR(dataset, dataset);
        VarList = strip(rowcatc("" + rowvec(inputs) + "','));
        VarList = substr(VarList, 1, nleng(VarList)-1);

        submit * /R;
            varnames <- c(&VarList)
            lm.MPG = glm(MPG_City ~., data=&dataset[varnames], family=&family)
            yhat = predict(lm.MPG,type="response")
```

```
        endsubmit;

        run ImportMatrixFromR(result, "yhat");    /* retrieve the result */
        return(result);
    finish;

    VarNames = {"Cylinders" "Model" "MPG_City" "Horsepower"};
    pred = GLMInR("Sashelp.cars", VarNames, "gaussian");
    print
pred; quit;
```

This example comprehensively demonstrates the techniques we introduced in section 2 and 3, including 1) transfer data set from SAS to R by calling ExportDataSetToR subroutine; 2) transfer IML variables as arguments in “submit /R” statement; 3) retrieve vector from R as the return of the module.

CONCLUSION

As an open source software which a variety of developers have contributed on, R provides statisticians with convenient access to state-of-the-art methodologies and implementations extensively and often fast. SAS PROC IML interfaces to R and makes all the R packages and functions available in the SAS environment. Data sets and variables get easily transferred between SAS and R using the methods as we have shown in the paper. Users can also create modules in PROC IML to repeat the routines of calling R functions in SAS without raising much code complexity.

CONTACT INFORMATION

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

Feng Liu
University of North Carolina at Chapel Hill
liufeng@email.unc.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.