PharmaSUG China 2016 - Paper 66

Hands-on Example of Matrix Operation in SAS with Different Approaches

Huan Cheng, Quintiles inc., Shanghai, China

ABSTRACT

It's not always that lucky the modeling used in the study is well developed by certain SAS procedure. Sometimes, we need to transform mathematical representation into executable SAS code. Most of the technical statistical literature prefers to use matrix algebra. Matrix algebra is also a simple and efficient way to do the calculation. When talking about the matrix operation in SAS, SAS/IML language is the first to come into mind. Besides SAS/IML, there are other ways to achieve the matrix operation. In the paper, first, it will give a brief overview of both PROC IML and PROC FCMP approach. Simple syntax and tips to use will be presented. Next, it will use computing the beta estimate of simple linear regression as example to illustrate how to use both IML and PROC FCMP to perform the operation. At last, it will have a little discussion about the two approaches to help users choose the proper method.

INTRODUCTION

For junior clinical programmers, most of the work is attributed to data manipulation. Matrix operation in SAS is perhaps untouched area. The paper aims to introduce how to perform the matrix calculation with different approach. Before discussing about the methods, a quick review of basic matrix algebra is presented.

QUICK REVIEW OF MATRIX ALGEBRA

A matrix is an $n \times p$ array of numbers. The integer n and p are the dimensions of the matrix. The row dimension is n; the column dimension is p. A vector is a special case of a matrix. A $n \times 1$ matrix is called a column vector, whereas a $1 \times n$ matrix is called a row vector. A 1×1 is called a scalar.

Some common operations of matrix are listed below.

- Matrix addition and subtraction
- Matrix multiplication
- Matrix transpose
- Matrix inverse
- Determinant of a matrix

Above definitions could be easily found in any algebra notebook. In most cases, the apparently complex formula is based on above basic operations. If you are able to decompose your formula, next section will introduce a simple procedure to achieve the programming.

AN OVERVIEW OF PROC FCMP

The SAS Function Compiler (FCMP) procedure is not new to most users. It provides the ability to build functions, CALL routines, and subroutines using DATA step syntax that is stored in a data set. Meanwhile, there is a special module in the procedure to provide a number of call routines for performing simple matrix operations. This is a relief for uses who do not have access to SAS/IML software.

All the operations listed in above section could find the corresponding routine function in PROC FCMP.

- ADDMATRIX Call Routine
- SUBSTRACTMATRIX CALL Routine
- MULT CALL Routine
- TRANSPOSE CALL Routine
- INV CALL Routine
- DET CALL Routine

Besides, PROC FCMP also provides the input data and output data function to enable the procedure to be processed by Data steps. READ_ARRAY function is to read arrays and WRITE_ARRAY is to write arrays to a data set.

Syntax of READ_ARRAY

```
rc = READ_ARRAY (data_set_name, array_variable);
```

Synatx of WRITE_ARRAY

rc = WRITE_ARRAY (data_set_name, array_variable);

data_set_name should be a literal string enclosed in quotation marks.

AN OVERVIEW OF PROC IML

The SAS/IML language is a programming language for high-level, matrix-vector computations. It shares many similarities with SAS DATA step. The language is not case sensitive and must end up with semicolon. There are some major differences as well. The most distinct one is in DATA step, the basic unit is observation, whereas in IML, the basic unit is matrix. DATA step will loop over observations in a data set; the SAS/IML will not. It's the often used for statistical computation rather than data manipulation. Compared with PROC FCMP, SAS/IML will enable you to write more concisely statistical expressions.

In SAS/IML program, all variables are matrix and it's no need to declare the type of the variable. Next section will bring the first glimpse of usage of PROC IML.

FIRST START OF A PROGRAM

Similar to DATA step, SAS/IML language is composed of statements. There are general four types of statements – Control statements, functions, call statements and subroutines and command statements. Below simple code illustrates the usage of each statement.

```
proc iml;
/* create a 3X3 matrix with 1 2 3  down the diagnol and
0 elsewhere*/
       x = j(3, 3, 0) ;
    do i = 1 to 3 ;
       x[i, i] = i ;
    end ;
/* create matrix Y, which is stored in row major order*/
       y={1 2 3 ,4 5 6, 7 8 9};
/* operation */
       * addition;
       z=x+y;
       *multiplication;
       a1=x*y;
       *transpose;
       a2=x`;
/* define a module*/
       start MyMod(a,b,c);
          a=sqrt(c);
          b=log(c);
       finish;
       run MyMod(S,L,y);

print z a1 a2 s;


quit;
```

DO ,END , START,FINISH,RUN belong to control statements, which direct the execution of the program and define DO groups or modules.

The matrix operation uses the function. There are six major functions – scalar function, matrix inquiry function, summary function, matrix reshaping function, linear algebraic function and statistical function. SAS/IML language supports most functions that are supported in SAS DATA step, like sqrt, log used in the code.

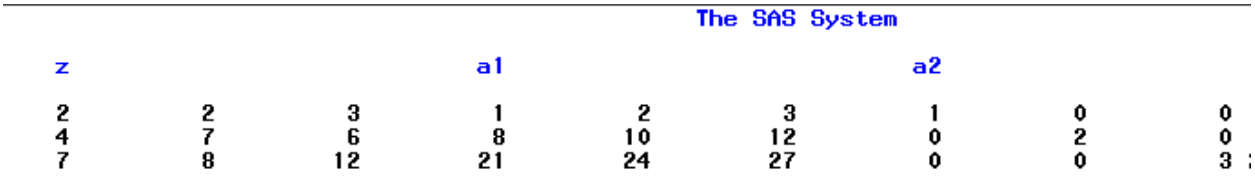
MyMod is a subroutine defined within PROC IML. The subroutine calculates the square root and log of elements provided in matrix c. The results are stored in matrix a and b.

PRINT is used to print the matrix or any messages. Figure 1 is a snapshot of the output from PRINT statement. The

matrix generated are listed.

Table 1 also tabulates the frequently used control or command statements in PROC IML. The data management command enables to manage the SAS data set with in IML environment. For example, you could import a SAS data set into the IML environment and also could export the matrix into a SAS data set.

Figure 1 Output of PRINT statement



AN EXAMPLE OF USING PROC FMCP AND PROC IML

After reviewing the basic knowledge of each procedure, an example will be presented in this section.

Suppose you are using regression analysis to descript the relationship between a child's weight and height and age. You may be interested in whether the height and age of child could predict the weight. The data could be found in the website of SAS user's guide. Unluckily, SAS cannot provide the OLS parameter estimates of multivariate regression (of course, lots of procedures exist actually), you are given below formula represented in matrix. We could solve the problem using both PROC FMCP and PROC IML.

$$\beta = \begin{bmatrix} b_0 \\ \cdots \\ b_{p-1} \end{bmatrix} = (X'X)^{-1}X'Y$$

Table 1 Frequently Used Statements

Control Statement	Command Statement	Data Management Command
DO, END	FREE	APPEND
GOTO, LINK	LOAD	CLOSE
IF-THEN/ELSE	MATTRIB	CREATE
PAUSE	PRINT	DELETE
QUITE	RESET	EDIT
RESUME	REMOVE	FIND
RETURN	SHOW	LIST
RUN	STORE	PURGE
START, FINISH		READ
STOP,ABORT		SETIN
		SETOUT
		SORT
		USE

PROC FCMP APPROACH

The code to calculate using PROC FCMP is presented below. Each operation of matrix will be performed in a call routine. It perfectly illustrates how the calculation is performed.

The data step is used to prepare the datasets for calculation. The intercept is also included in the model, so x0 represents the constant vector. X1 and x2 represents the height and weight. Figure 2 is the data set snapshot.

In the PROC FCMP step, the matrix must be declared before use. The dimensions of row and column must be clearly specified. This is different from SAS/IML environment, in which the variable is supposed as matrix and don't need to specify the type.

```
data x (keep=x0 x1 x2) y(keep=y);
       retain y x0 x1 x2;
       set class end=eof;
       x0=1;
       y=weight;
       x1=height;
       x2=age;
       if eof then call symputx('nrow',_N_);
       keep y x1 x2 x0;
run;



proc fcmp;

       ** read the X matrix and Y vector;
array x[&nrow.,3]/nosymbols;
rc=read_array('x',x);
array y[&nrow.,1]/nosymbols;
rc=read_array('y',y);
       **t(X);
array x_t[3,&nrow.]/nosymbols;
call transpose (x,x_t);
       **t(X)X;
array xx_t[3,3]/nosymbols;
call mult(x_t,x,xx_t);
       ** inv(t(X)X);
array xx_t_inv[3,3]/nosymbols;
call inv(xx_t,xx_t_inv);
       **inv(t(X)X)X;
array xx_t_inv_x[3,&nrow.]/nosymbols;
call mult(xx_t_inv,x_t,xx_t_inv_x);
       **inv(t(X)X)XY;
array beta[3,1]/nosymbols;
call mult(xx_t_inv_x,y,beta);
       **output the dataset;
rc=write_array('beta',beta);

run;
```

Figure 2

	x0	x1	x2
	1	69	14
	1	56.5	13
	1	65.3	13
	1	62.8	14
	1	63.5	14
	1	57.3	12
	1	59.8	12
	1	62.5	15
	1	62.5	13
	1	59	12
	1	51.3	11
	1	64.3	14

Figure 3 is the dataset – beta generated by the procedure. Row1, Row 2 and Row3 correspond to b0, b1 and b2 in the vector.

Figure 3

	beta1
1	-141.2237635
2	3.5970265113
3	1.2783925135

PROC IML APPROACH

The below code is composed of control statements and operation functions. USE and READ statements import the SAS data sets and variable into the IML environment. CREATE and APPEND export the matrix into SAS data sets. Two solutions to get the final beta estimates are presented. The first one uses INV function, which is not the recommended one. SOLVE function is the more efficient one to solve the normal equation.

```
proc iml;
  * Read data into IML ;
  use class;
  read all var {weight } into y ;
  read all var {height age } into x ;
  nx=nrow(x);
  ** create the intercept vector;
  b0=repeat({1},nx,1);
  **concatenate;
  x1=b0||x;
  p1=x1`*x1;
  p2=x1`*y;
  **solution 1;
  beta=inv(p1)*p2;
  *solution 2;
  beta1=solve(p1,p2);
  ** write result into sas data;
  create beta1 from beta ;
  append from beta;
  close beta1;

quit;
```

The estimates from PROC IML are exactly the same as given by PROC FCMP. If you are interested in the numbers estimated by the embedded procedure in SAS. Figure 3 is the parameter estimates from PROC REG.

Figure 4

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-141.22376	33.38309	-4.23	0.0006
Height	1	3.59703	0.90546	3.97	0.0011
Age	1	1.27839	3.11010	0.41	0.6865

EXNTESION IN PROC IML

There are lots of functions in the SAS/IML environment. For example, you can generate a list of random numbers and perform calculation based on it within the environment. If using PROC FCMP, you need to first create a dataset which contains the random numbers and then import the dataset in PROC FCMP.

Suppose you want to generate a series of normally distributed numbers and calculate the mean, the 95 percentile and 5 percentile. Below code performs the computation.

CALL RANDGEN corresponds to the RAND function in the DATA step. CALL SORT sorts the X matrix like what PROC SORT does. Below code could be programmed using DATA steps, but it'll be lengthy.

```
%let K=1000000;/*number of simulation*/
%let alpha=0.1;
proc IML;
call randseed(123); /* set random number seed */
X = j(&K.,1); /* allocate the X vector */
call randgen(X,"Normal");
create X var{X}; append;
close X;
call sort(X);
LOW=X[round(&k.*(&alpha./2))]; print low;
UP=X[&K.-round(&k.*(&alpha./2))]; print up;
MEAN=mean(X);
**write to UPLOW dataset;
create uplow var{low mean up};append;
quit;
```

Though we could mimic DATA step processing, there are still some differences between IML and DATA steps.

- With SAS/IML, CREATE statement instead of DATA statement to start. APPEND statement is used to output an observation, while in DATA step, the observations could output automatically.
- CLOSE statement must be specified to close the data set. DATA step could do this automatically.

In terms of the data processing capability, DATA step have the advantage of simplicity. However, SAS/IML has more flexibility.

CONCLUSION

The regression example is quite simple, but it's enough to expose the advantages or disadvantages of each method. SAS/IML is developed to solve matrix computation, so it's the first choice without doubt if you have access to the software. PROC FCMP module provides the basic operation call modules and it's the few choice if you only have base SAS access. Obviously, in terms of matrix computation capability, the PROC FCMP module is not that efficient

and succinct compared with SAS/IML. Sarfaraz and Binoy (2010) provide a method to perform the matrix operation via DATA steps. Compared with the lengthy code in the paper, PROC FCMP seems friendlier.

REFERENCES

Wicklin, Rick. (2013). “Getting Started with the SAS/IML® Language” Available at https://support.sas.com/resources/papers/proceedings13/144-2013.pdf

Sayyed , Sarfaraz and Varghese, Binoy. (2010). “Do you want to do Matrix Calculations with Base SAS Software without SAS/IML?” Available at http://www.lexjansen.com/pharmasug/2010/CC/CC15.pdf

RECOMMENDED READING

- Base SAS® Procedures Guide
- SAS® For Dummies®
- SAS/IML(R) 9.3 User's Guide

CONTACT INFORMATION

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

Name: Huan Cheng
Enterprise: Quintiles
Address: 5F Building A, Fenglin International Tower, 388 Fenglin Road, Xuhui District
City, State ZIP: Shanghai, 200032
Work Phone: + 862124228754
E-mail: hcheng1118@hotmail.com
Web: https://www.linkedin.com/in/huancheng