Matrix algebra

Matrix algebra is very prevalently used in Statistics because it provides representations of models and computations in a much simpler manner than without its use. The purpose of this section is to provide the basics of commonly used matrix algebra operations in SAS. The program used in this section is basic_matrix_algebra.sas.

Basics

A matrix is simply a rectangular arrangement of elements in rows and columns. For example, consider the matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Letters that represent matrices are always bolded or written with a line underneath as $\underline{\mathrm{A}}$ when bolding is not possible, like on a chalkboard. The symbolic elements of a matrix are written with a subscript for the row and column numbers (row, column). The dimension of the matrix is its number of rows and columns. For the above matrix, the dimension is 2×3.

The SAS procedure `proc iml` enables the use of matrix algebra (IML stands for "interactive matrix language"). To illustrate its use, consider the following two matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} -1 & 10 & -1 \\ 5 & 5 & 8 \end{bmatrix}$$

Below is how we can enter these matrices into SAS and print them.

```
title1 "Chris Bilder, STAT 850";
proc iml;
```

```
A = {1 2 3,
     4 5 6};
print A;

B = {-1  10 -1,
      5   5  8};
print B;

print A, B;
```

Chris Bilder, STAT 850

A		
1	2	3
4	5	6

B		
-1	10	-1
5	5	8

A		
1	2	3
4	5	6

B		
-1	10	-1
5	5	8

Elements of the matrix can be extracted by specifying a row and column number:

```
temp = A[1,1];
print temp;
temp = A[1,];
print temp;
temp = A[,1:2];
print temp;
```

```
temp = A[,1];
print temp;
```

temp
1

temp		
1	2	3

temp	
1	2
4	5

temp
1
4

The extra step of saving a result is needed prior to printing, so a statement like `print A[1,1]` does not work.

Below are examples of adding and subtracting with matrices:

```
C = A + B;
print C;
D = A - B;
print D;
```

C		
0	12	2
9	10	14

D		
2	-8	4
-1	0	-2

Matrix multiplication

Consider the following two matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 3 & 0 \\ 1 & 2 \\ 0 & 1 \end{bmatrix}$$

Below is how matrix multiplication can be performed with these matrices.

```
A = {1 2 3,
     4 5 6};
B = {3 0,
     1 2,
     0 1};
print A, B;

E = A*B;
print E;
F = B*A;
print F;
A3 = A*3;
print A3;

*Elementwise multiplication;
C = A#A;
print C;
```

A		
1	2	3
4	5	6

B	
3	0
1	2
0	1

E	
5	7
17	16

F		
3	6	9
9	12	15
4	5	6

A3		
3	6	9
12	15	18

C		
1	4	9
16	25	36

Inverse of a matrix

The `inv()` function calculates an inverse:

```
A = {1 2,
     3 4};
```

```
print A;

Ainv = inv(A);
print Ainv;
check = A*Ainv;
print check;
```

A	
1	2
3	4

Ainv	
-2	1
1.5	-0.5

check	
1	1.11E-16
0	1

More matrix operations

Commonly used operations with matrices include finding the diagonal elements, finding determinants, taking a transpose of a matrix, and forming a diagonal matrix:

```
A = {1 2,
     3 4};
print A;
Adiag = diag(A);
print Adiag;

*Determinant;
Adet = det(A);

*Tranpose;
Atran = t(A); *A‘ also works;
```

```
print Adet, Atran;

*Create diagonal matrix;
*  Create 1x5 vector and use elementwise multiplication with a
   5x5 identity matrix;
DiagMat = {1 2 3 4 5}#I(5);
print DiagMat;
```

A	
1	2
3	4

Adiag	
1	0
0	4

Adet
-2

Atran	
1	3
2	4

DiagMat				
1	0	0	0	0
0	2	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

Eigenvalues

Eigenvalues and eigenvectors are found by using the `eigval()` and `eigvec()` functions, respectively.

```
A = {1   0.5,
     0.5 1.25};
print A;

lambda = eigval(A);
lambdavec = eigvec(A);
print lambda, lambdavec;

check1left = A*lambdavec[,1];
check1right = lambda[1]*lambdavec[,1];
print check1left, check1right;

check2left = A*lambdavec[,2];
check2right = lambda[2]*lambdavec[,2];
print check2left, check2right;
```

A	
1	0.5
0.5	1.25

lambda
1.6403882
0.6096118

lambdavec	
0.6154122	0.7882054
0.7882054	-0.615412

check1left
1.0095149
1.2929629

check1right
1.0095149
1.2929629

check2left
0.4804993
-0.375163

check2right
0.4804993
-0.375163

To verify the eigenvectors, I used the relationship that eigenvectors of a matrix $\mathbf{A}$ satisfy $\mathbf{Ab} = \lambda\mathbf{b}$ where $\mathbf{b}$ is an eigenvector.

Please remember that eigenvectors are not unique! Eigenvectors in SAS are scaled to have a length of 1. Still, there is more than one vector that can have a length of 1.

SAS data sets and `proc iml`

A typical work flow in SAS is to use SAS data steps and procedures as much as possible and then use `proc iml` for tasks that are not available in procedures. As one might expect, this is what occurs a lot for statistical research.

Below is an example of how to import a data set already in SAS into `proc iml` and then to perform basic calculations to estimate the regression parameters of a simple linear regression model.

```
data set1;
  infile "C:\data\gpa.csv" firstobs=2 delimiter=",";
  input HS College;
run;

proc iml;

  use set1;
    read all var{HS} into Xpart;
    read all var{College} into Y;

  *Number of observations;
  n = nrow(Xpart);
  *nx1 vector of 1's;
  vec1 = J(n, 1, 1);
  *Horizontal concatenate;
  X = vec1 || Xpart;
  *print X, Y;
  betahat = inv(t(X)*X) * t(X) * Y;
  print betahat;

  *Shows how to read in all variables at same time;
  read all into XY;
  *print XY;
```

```
quit;
```

betahat
1.0868795
0.6124941

Notice the use of the functions `nrow()` and `J()`. There are many other functions available! Please take a look at the SAS help available at

Using R in SAS

Most statistical software packages now include ways to use R within them. For SAS, this can be done within `proc iml`. To enable the use of R, one needs to start SAS with the additional option of `-RLANG`. For example, I have a link on my desktop which uses the following target:

```
"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -CONFIG
  "C:\Program Files\SASHome\SASFoundation\9.4\nls\en\sasv9.cfg"
```

```
-RLANG
```

All parts of this target are the default except the new `-RLANG` portion.

R code is run directly from within `proc iml` by enclosing it within

```
submit / R;
```

and

```
endsubmit;
```

statements. Mostly all types of R code work from within SAS. Below are a few examples:

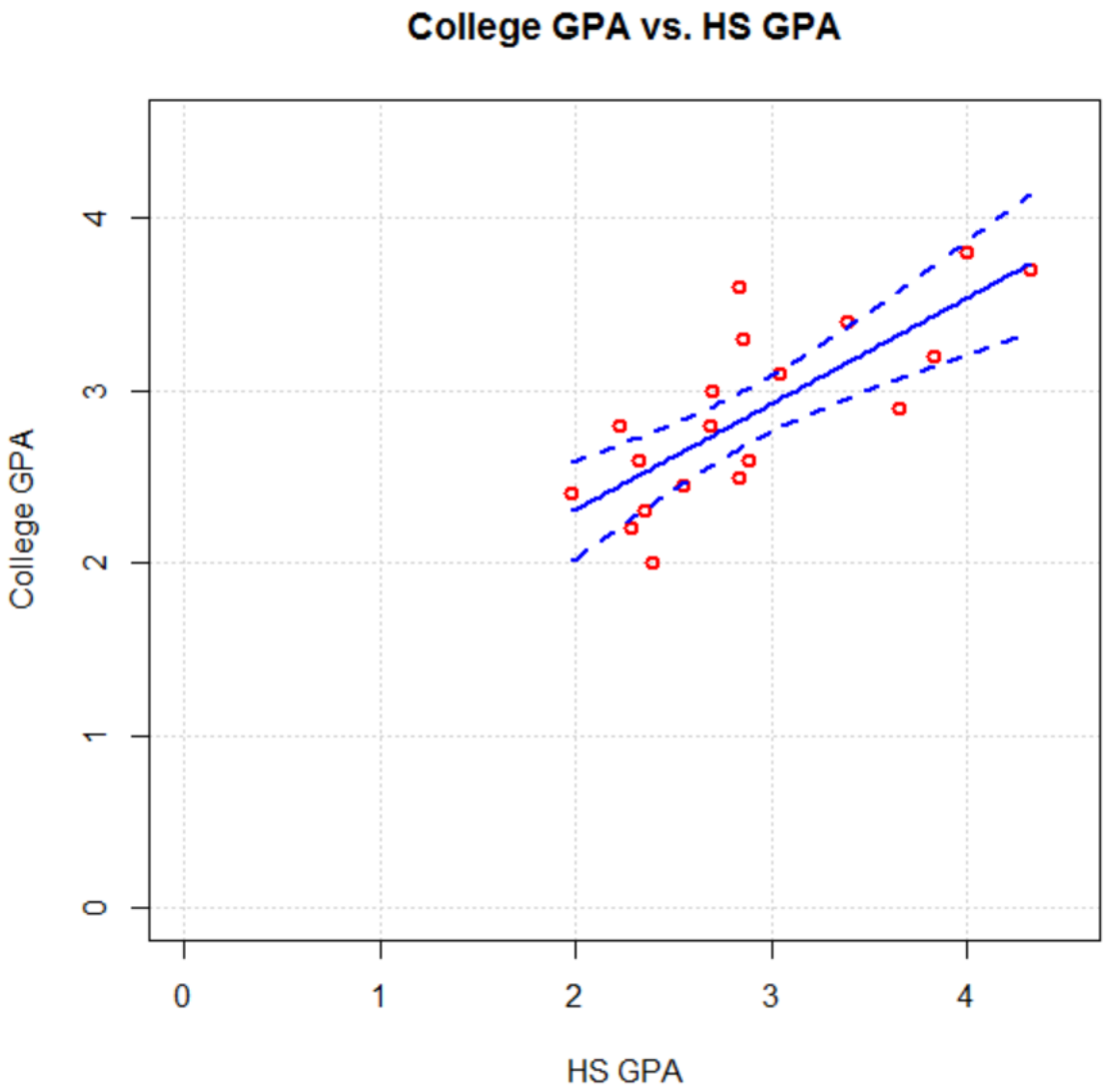
```
proc iml;

  *Simple example;
  submit / R;
    x <- 1
    x
  endsubmit;

  *Use a SAS data set in R;
  run ExportDataSetToR("set1", "Rset1");

  submit / R;
    head(Rset1)
    mod.fit <- lm(formula = College ~ HS, data = Rset1)
    names(mod.fit)
    mod.fit$coefficients
    #summary(mod.fit) #Does not work!

    plot(x = Rset1$HS, y = Rset1$College, xlab = "HS GPA", ylab
      = "College GPA", main = "College GPA vs. HS GPA", xlim =
      c(0,4.5), ylim = c(0,4.5), col = "red", pch = 1, cex =
      1.0, lwd = 2.0, panel.first = grid())
    curve(expr = predict(object = mod.fit, newdata =
      data.frame(HS = x)), col= "blue", add = TRUE, lwd = 2,
      xlim = c(min(Rset1$HS), max(Rset1$HS)))
```

```
    curve(expr = predict(object = mod.fit, newdata =
      data.frame(HS = x), interval = "confidence", level =
      0.95)[,2], col= "blue", add = TRUE, lwd = 2, xlim =
      c(min(Rset1$HS), max(Rset1$HS)), lty = "dashed")
    curve(expr = predict(object = mod.fit, newdata =
      data.frame(HS = x), interval = "confidence", level =
      0.95)[,3], col= "blue", add = TRUE, lwd = 2, xlim =
      c(min(Rset1$HS), max(Rset1$HS)), lty = "dashed")

  endsubmit;

  *Use R vector in SAS;
  run ImportMatrixFromR(betahats, "mod.fit$coefficients");
  print betahats;

  *Use R data frame outside of proc iml;
  run ImportDataSetFromR("set2", "Rset1");

quit;

proc print data = set2(obs=5);
run;
```

```
[1] 1
```

```
    HS College
1 3.04    3.10
2 2.35    2.30
3 2.70    3.00
4 2.55    2.45
5 2.83    2.50
6 4.32    3.70
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
(Intercept)          HS
  1.0868795   0.6124941
```

betahats
1.0868795
0.6124941



Obs	HS	College
1	3.04	3.10
2	2.35	2.30
3	2.70	3.00
4	2.55	2.45
5	2.83	2.50

Comments:

- Multiple `submit / R` and `endsubmit` pairs can be made within one `proc iml` and `quit` set of code. R objects created by an initial `submit / R` and `endsubmit` sequence are available for other sequences.

- SAS data sets and matrices can be used in R, and R objects can be used in SAS. Please see the examples above and the additional examples given in the program. With these import and export functions, it is important to use the quote symbols correctly–some data sets or objects have quotes while others do not.
- R plots are opened in an R graphics window. This window will be closed after the `quit` statement for `proc iml` is executed.

The SAS blog entry at `http://blogs.sas.com/content/iml/2013/11/25/twelve-advantages-to-calling-r-from-the-sasim.html` gives some good advice about using SAS and R together.

Ending `proc iml`

After executing statements in `proc iml`, you will notice that SAS indicates that `proc iml` is still running. In order to fully exit out of the procedure, a `quit` statement needs to be given.