X.3 SAS/IML to GAUSS の基礎　　ver. 0.1

SAS は広大な分野領域を含む総合言語です。そのなかに GAUSS と同じような行列操作と関数作成にすぐれている IML という PROC 部分があります。ここでは、SAS の一般的なファイル入力と IML に特化した行列操作からの GAUSS へのコンバートを扱います。もちろん、IML 以外でも SAS では GAUSS のようなことはできる広大な言語体系です。

コメント行

SAS	GAUSS
/* This is a message. */	/* This is a message. */

コメント行は、SAS も GAUSS もともに同じ扱いです。行の途中から始まっても、複数行にまたがっていても、/*　*/で囲まれた部分は実際の計算実行では無視されます。

画面表示

SAS	GAUSS
PRINT 'This is a message.';	print "This is a message.";
PRINT Y;	print y;

SAS では文字列は'　'で包んで使います。したがって、画面表示の PRINT の際も同様に PRINT 文のあとにその引用符に囲まれた文字を置きます。変数の中味の表示の場合にも GAUSS とまったく同様に扱います。命令文の最後にセミコロン；をつけるのも同じです。

外部ファイルの読込み

SAS	GAUSS
DATA DATAF1;	
INFILE 'D:DATAFILE1.TXT';	load dataf1[29,2]=d:datafile1.txt;
INPUT Y X2;	y=dataf1[.,1]; x2=dataf1[.,2];

SAS ではファイルの読込みをする場合には DATA ステップで名前をつけた上で（ここでは DATAF1 と命名）、INFILE を用いて引用符内のファイルを読込みます。引用符は、シングルでもダブルでもどちらでもかまいません。このことは SAS の文字列の扱い全般に言えることです。それを INPUT 命令でそれぞれの列を変数 Y と X2 という名前をつけます。一方 GAUSS では、load 命令でダブルの引用符に囲まれたファイル名を読込みます。なお、SAS も GAUSS もディレクトリが何層かにわたってある場合にはバックスラッシュに相当する半角の￥のマークまたは、その代用としてのスラッシュ/を用いて区切ります。GAUSS では、いったん読込んだ行列をさらに行のワイルドカードのしるしドットを用いて、１行目はｙ、２行目はｘ２などというように変数名をつけます。

ファイル内部入力

```
================================================================
              SAS                                GAUSS
================================================================
        DATA DATAFILE1;
        INPUT Y X2;
        CARDS;                          load datafile1[4,2]=
        2 4                             2 4
        3 6                             3 6
        4 8                             4 8
        5 11                            5 11
        ;                               ;
        PROC IML;
        USE DATAFILE1;
        READ ALL VAR{Y} INTO Y;          y=datafile1[.,1];
        READ ALL VAR{X2} INTO X2;       x2=datafile1[.,2];
================================================================
```

データ数がそれほど多くない場合や、カットアンドペーストでワークスペースにもってこれる場合には、外部ファイルの読込みではなくて、プログラムファイル内部でデータを入力することも可能です。SAS の場合には、DATA ステップで、データ全体に内部での名前をつけ、INPUT 命令で各変数に名前を割り振った後で、おなじみの CARDS 機能を用います。SAS では行の概念が CARDS 機能にはあります。それぞれが１行になっています。そのデータを用いるには、ほかの PROC ステップで、USE 命令で DATA ステップでつけた名前を持ってきて使える状態にします。なお、READ ALL;でも事足りますが、さらに変数名を IML 内で何か別のものに割り振る場合には、READ ALL VAR{Y} INTO Y;などとします。一方、GAUSS では、行の概念が存在しないので、通常は datafile1={1 2,3 4, …}など

と｛　｝括弧に数値を入れて、カンマで区切ってそこまでを１行とします。しかしながら敢えて、カットアンドペーストで持ってきたスペースだけで区切られたデータを読込むには let 命令を使います。配列を指定した変数名を let 命令でイコールサインでつなげると、べた読みできます。配列が指定されないと GAUSS では行の概念がないので１行として読込まれますが、配列を指定して無理やり let で整形してやると SAS の CARDS 機能と同じようなことができます。各変数への割り当ては、外部ファイルの読込みの時と同様に行ないます。もちろん、DATA ステップなしに、PROC IML 内で、datafile1={1 2,3 4,…};などというふうに GAUSS ライクに行列の要素を入力することも可能ですが、この場合、そのほかの PROC で利用できない IML 内部だけの変数になります。

プロット

SAS	GAUSS
PROC PLOT DATA=DATAFILE1;	library pgraph; graphset;
PLOT Y*X2;	
TITLE 'XY PLOT';	title("xy plot");
TITLE2 'Y';	ylabel("y");
TITLE3 'X2';	xlabel("x2");
	_plctrl = -1;
	xy(x,y);

SAS では、あらかじめ DATA ステップでつけられ名前のデータを、その PROC で指定して動作をさせます。ここでは、PLOT をさせるのにデータを DATA＝DATAFILE１と指定しています。PLOT では、＊印でｙ軸とｘ軸の変数を分けます。グラフに文字を入れるのには、TITLE、TITLE2、TITLE3 で引用符で包んで指定します。一方、GAUSS では、ライブラリを呼び出してこないとグラフは書けないようになっています。ここでは、pgraph のライブラリを呼び出して、graphset;でそのグローバル変数を初期化してから、xy グラフを描かせています。GAUSS ではまず、ｘｙ命令よりも前に何らかのオプションを設定しなければ無視されてしまいます。ここではタイトルとｘとｙのラベルを先に指定します。なお、GAUSS ではｘ、ｙという順番です。また、xy のプロットのデフォルトは時系列などの線を結ぶグラフなので、線を取り除くには、_plctrl = -1;とプロットコントロールの値を－１にする必要があります。

OLS

SAS	GAUSS
PROC REG DATA=DATAFILE1; MODEL Y=X2;	call ols(0,y,x); or call lse(y,x);

SAS では、DATA ステップでデータを作成したものを、ほかの PROC ステップにその都度持ち込んで何かの処理を行なう形をとります。ここでは、あらかじめ DAYAFILE1 というものが DATA ステップで用意されているとして、それを PROC REG のなかで、DATA＝で指定してやります。それを MODEL Y=X2;で定数項つきで OLS 回帰させています。一方、GAUSS では、データステップというものは存在しなくて、上の例の場合、ｙとｘという行列または列ベクトルが指定されているとして、それを procedure を呼び出してインプット変数として代入して処理をします。組込みされている OLS のような内部 procedure もあるでしょうし、多くは自作の lse のような procedure をプログラム後半で呼び出します。

IML 内だけの行列入力

SAS	GAUSS
X={1 2,3 4,5 6}; Z={‘X1’ ‘X2’, ‘X3’ ‘X4’};	x={1 2,3 4,5 6}; z={“X1” “X2”,”X3” “X4”};

SAS でも GAUSS でも行列の要素の入力方法は同じです。カンマまでが１行になります。文字列の場合には引用符に包みます。ただし、GAUSS では数値と文字列は厳格に分離して扱われるので、例えば上のｚの２×２の文字列行列を表示するには$z を print する必要がありますし、加工したりする場合には、演算子の前に＄のしるしをつけます。SAS/IML では上のように行列を扱えますが、これはデータとはなっていません。データとしてほかの PROC ステップに使えるようにするには上述のように DATA ステップが必要になります。

IML での配列表記

SAS	GAUSS
x[2,1]	x[2,1]

配列表記は SAS も GAUSS もまったく同じです。サブ行列も、コロンを用いて x[1:2,2:3] などとすることで SAS も GAUSS も同様になります。四角括弧[　]の中に書きます。ただし、その操作については、次のように、かなりのところが違ってきます。

IML での配列操作

SAS	GAUSS
x[,1]	x[.,1]
x[+,]	sumc(x)'
x[+,+]	sumc(sumc(x))
x[##,]	sumc(x^2)'

SAS では、ワイルドカードに行または列をする場合、何もつけません。GAUSS ではワイルドカードはドットマークです。SAS では、配列の四角括弧の中の＋マークは合計を表します。上の x[+,]の場合、行のところに＋があるので、行の中でをすべて足し合わ流という意味で、列ごとの合計が得られます。一方、GAUSS にはそのような配列表記は存在しなくて、組込み関数 sumc を使います。なお、sumc で出てくる答えは１列の縦ベクトルになります。SAS では、行も列も両方に＋を置くと、全体の合計になります。GAUSS では、上の sumc を二重にすることで解決します。GAUSS ではこのような二重の関数表記ができるように故意に答えは列ベクトルになるような仕様になっています。SAS では、##とすることで、上の場合、列ごとの２乗和が計算できます。なお x[,:]などとすると平均値が得られます。

IML での行列演算

SAS	GAUSS
z=y**2;	z=y^2;
z=y#x2;	z=y.*x2;
z=y@z2;	z=y.*.z2;
c=a<>b; c=a><b;	
m=x`*x;	m=x'x;

SAS と GAUSS は演算記号はほぼ同じだが、細かいところで若干異なる。特に要素対要素の扱いが違う。SAS ではべき乗計算では＊＊を用いるのに対して、GAUSS では ^ を用い

る。SAS でべき乗のかけ合わせる側がスケラーでない場合には＃＃を用いる。＃のマークは要素対要素の掛け算にも使われる。なお、SAS では＾のマークに論理で NOT の意味がある。クロネッカー積は、SAS では@を、GAUSS では.*.となる。SAS 特有の演算としては例えば、a={1 2 1}; b={2 1 0};とする時、a<>b は２つの行列の大きな要素のもので{2 2 1}となる一方、a><b は２つの行列の小さな要素で{1 1 0}となる。逆行列の計算では注意を要する。GAUSS では'で転置を作れて、その場合に限って＊のマークなしにかけ合わせることができる。しかしながら、SAS では、転置は`のマークを使い、しかも＊のマークは行列をかけあわせる場合省略はできません。

IML 上の組込み関数

SAS	GAUSS
N=NROW(X);	n=rows(x);
K=NCOL(X);	k=cols(x);
CS=CUSUM(X);	cs=cusumc(x);
XMAX=MAX(X);	xmax=maxc(x);
XMIN=MIN(X);	xmin=minc(x);
LNX=LOG(X);	lnx=ln(x);

SAS では、行の数および列の数を求めるのに NROW と NCOL を使います。GAUSS では、rows と cols です。SAS にも CUSUM や MAX それに MIN がありますが、これらは行列のすべての要素に対するものです。決して、GAUSS のように列ごとに計算するものとはなっていませんので注意してください。逆行列は同じ INV です。なお、逆行列は**.5 とすることによって SAS では求まりますが、GAUSS ではそれぞれの要素の逆数を計算してしまいます。SAS では LOG は自然対数のことです。GAUSS では自然対数には ln を使います。その他行列の操作に関する関数が SAS には数多く備え付けられています。ただし統計に関する関数は、一度 IML を抜けて PROC ステップで別の PROC をその都度呼び出します。

IML での行列マージ

SAS	GAUSS
c=a | | b;	c=a~b;
c=a//b;	c=a | b;

注意してほしいのが、行列の水平方向および垂直方向のマージである。SAS では水平方向には| |を用い、垂直方向には//を用いる。GAUSS の感覚から言えば逆のような気もしないでもないが、GAUSS では水平方向には～、垂直方向には｜を用います。

シークエンス数列および零行列

SAS	GAUSS
I=1:10;	i=seqa(1,11,1)';
J(3,3)	ones(3,3)
J(4,2,0)	zeros(4,2)

SAS では数列を作る時には、コロンを用いて始点から終点までを指定する。何も指定しない場合にはステップ１または－１で増えるか減る。一方、GAUSS では seqa を用いて、この場合、１から 11 個、ステップ１刻み というふうにしていする。距離とポールの数の問題のように、10 の間にポールは 11 本だから注意する必要がある。GUASS で ones と zeros に相当する零行列または要素が１ばかりの行列を作るには、SAS では J 行列を用いる。この場合、最後の３つ目の要素の数は、埋めるべき要素の値になる。J(4,2,0)の場合、0 ばからの要素で埋めることになる。３つ目の数を省略すると J 行列は１ばかりで埋める行列であるデフォルトになっている。

Do ループ

SAS	GAUSS
I=1;	i=1;
DO WHILE (I<=10);	do while i<=10;
PRINT I;	print i;
I=I+1;	i=i+1;
END;	endo;

SAS には GAUSS と同様に DO ループを行なえます。ただし、最後は END;となる。また、同じ文法で、SAS も GAUSS も DO UNTIL のループもできます。DO UNTIL の場合には SAS でも GAUSS でも同様に I＜＝10 の NOT の I が 10 を超えるまで、つまり I>10 までを設定すると上と同じ回数だけループが回って同じ結果になります。

数列 Do ループ

SAS	GAUSS
DO I=1 TO 10 BY 1;	for i (1,10,1);
PRINT I;	print i;
END;	enfor;

SAS でも数列で DO ループをまわせるようになっている。この場合、ステップは１で１の場合には BY 1 に相当する部分は省略できる。最後は END である。一方、GAUSS では、for ループを用いて、ｉが１から 10 まで１ずつ増加するループが endfor の間を回ることになる。GAUSS の場合ステップ幅は１であっても省略はできない。

条件分岐

SAS	GAUSS
IF B>0	if b>0;
THEN Y=1;	y=1;
ELSE	else;
Y=0;	y=0;
(END;)	endif;

SAS では、IF 条件分岐では then を使って、else の直前までを実行します。GAUSS の場合、最後の else;から endif;までがそれ以外の場合の実行部分になりますが、SAS の場合明らかな場合にはおしまいは省略されるか、END;をつけます。GAUSS では、最後までがそれ以外の else に相当してようなかろうといかなる場合にも endif;文は if 文に必要になる。

モジュール

SAS	GAUSS
	z=mycalc(2,3);
	print z;
D=0;	d=0;
START MYCALC(A,B);	proc mycalc(a.b);

```
    C=A**B+2;                          local c;
    D=D+1;                              c=a**b+2;
FINISH MYCALC;                          d=d+1;
                                        retp(c);
                                    endp;
RUN MYCALC(2,3);
PRINT C;
```

==========

SAS では、IML の中では START に始まって FINISH で終わるモジュールが書けます。上の例ではリターン文を省略した例ですが、これでも計算はできます。START 文の後の名前はモジュール名で、丸括弧の中にインプット変数が入っています。それを後で、RUN または CALL 文によって呼び出して使います。明示的なローカル変数の表示はありませんが、上の例では D はグローバルで、モジュールの中で１つ増して再定義されています。C はローカル変数に相当する部分です。このモジュールを呼び出した時にのみ有効です。一方、GAUSS では、ローカル宣言文が必要です。リターン文も、内部で出力する場合以外には必要になってきます。この場合、ローカル変数 c がリターンとして返されます。それを冒頭で呼び出す形となっています。