X.4 SHAZAM to GAUSS の基礎　　ver. 0.1

SHAZAM は計量言語上もっともすっきりとした文法で書かれ、コマンドを使ったパッケージ計算とコードライティングによる計算が有機的に統合されている言語です。すでに 90 年代初頭までに基本的なところはできあがっており、マニュアルは計量言語の中で最もクオリティーの高いものになっています。

コメント行

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

SHAZAM は FORTRAN 系ベースの言語と言われているので、コメントは行単位で、その先頭に＊のマークをつけることによって表します。一方、GAUSS には行の概念はないので行のどこかに/*　*/で囲まれた部分を置くと複数行にまたがっていようとコメントとして実行上無視されます。

画面表示

SHAZAM	GAUSS
PRINT Y	print y;

SHAZAM では GAUSS と同様に、丸括弧なしに変数を置くタイプの言語で、PRINT 文で画面表示します。ただし行の概念はないので、SHAZAM のプログラムにはセミコロン；はつけません。

外部ファイルの読込み

SHAZAM	GAUSS
[省略可] SAMPLE 1 29	
READ (d:datafile1.txt) Y X2	load dataf1[29,2]=d:datafile1.txt; y=dataf1[.,1]; x2=dataf1[.,2];

SHAZAM では READ 文で丸括弧内の外部ファイルを読込みます。その後に、それぞれの列のシリーズの名前を列の数だけつけます。上から何行かとばす場合には、/SKIPLINES = のオプションをさらにその後ろにつけます。変数名とスペースを置かずに書かれていることも多いので割り算か何かと混同しないでください。また、WINDOWS ではなくてメインフレームか何かを対象に書かれているプログラムの場合、FILE 文でファイル名が装置番号という数字に割り当てられて READ(数字)などという形になることもありますがびっくりしないでください。FORTRAN 風に割り当てられているにすぎません。全部の行利用には省略もできますが通常 READ 文の前には SAMPLE 文といって系列のサンプル数がどこからどこまでかを指定する文があります。例えば、SAMPLE 1 29 などとなっています。なお READ 文ではなく FILE 文で入力する方法も別にあります。一方、GAUSS では、load 文で行列に縦横の配列数を指定した上で代入します。各変数に分ける作業は、行列として列ごとにワイルドカードのドットのしるしを使って割り当てます。

ファイル内部入力

SHAZAM	GAUSS
DIM X 4 4	
READ X /ROWS=3 COLS=3	let x[3,3]=
1 2 3	1 2 3
4 5 6	4 5 6
7 8 9	7 8 9
	;

SAS の CARDS 機能のように内部でデータを読込むには、SHAZAM では、READ 機能を用いて、その後に変数名とスラッシュの後に行と列の数を指定した上で、その行のあとに直接データを何行かにわたって書き入れます。もし、DIM 文がその前についていたならば（通常は必要ありませんが)、変数のディメンションを強制的に上書き指定することができます。DIM X 2 2 とすれば行列 X のサブ行列が２行２列で作成されます。上の場合４行４列というふうにその後で READ 文で読込むもとの行列よりも大きくなっています。その場合には、要素が存在しないところにはすべて０が入ります。一方、GAUSS では｛　｝括弧の中にデータを書く方法もありますが、ディメンションを指定する場合には、let 文を使って、その後に配列つきの変数を置きます。なお、その場合、GAUSS では 1 2 3 4 5 6 7 8 9 などとべた書きにしてもかまいません。それを強制的に四角括弧内の配列に整形することになります。最後には、必ずセミコロン；が必要です。

プロット

SHAZAM	GAUSS
	library pgraph; graphset;
PLOT Y X2	xy(x2,y);

SHAZAM では、PLOT 文でｙ軸側の変数（複数)、最後にｘ軸側の変数を置いてプロット表示させます。TIME に対して自動表示させる場合には、従属変数の部分は指定しなくて

PLOT X2 /TIME

とします。一方、GAUSS はグラフ機能は外部化されていて、pgraph というライブラリを呼び出してきて、graphset;でそのグローバル変数設定を初期化してから描画命令 xy を置きます。GAUSS の xy グラフの場合もｙ軸側そしてｘ軸側の順です。GAUSS では、ｘ軸側が複数ある場合には変数を水平方向にマージしてからｘに相当する部分に代入します。

OLS

SHAZAM	GAUSS
OLS Y X2	call ols(0,y,x);　or　call lse(y,x);
?OLS Y X2	

SHAZAM では OLS コマンドの後に従属変数、独立変数の順で変数を置きます。一方、GAUSS では通常行列計算で行ないますが、簡易的に ols コマンドが備え付けられていますし、プログラムの節で後述する lse などの proc を自作して、プログラム後半に置いてそれを前半部で呼び出す形もとれます。なお、SHAZAM では OLS コマンドにともなって、

$N	データの個数
$K	係数の個数
$DF	自由度
$R2	決定係数
$SSE	SSE
$LLF	Log-Likelihood の値

などの＄が先頭についた一時変数がその都度内部でできていて、それを直接その後の計算で用いていきます。また、　　OLS Y X2 /RESID=R PREDICT=PY

などとしてオプションとして残差や推定値を RESID=や PREDICT=の後にくる変数として

指定して、その後の計算で利用することも多々行なわれます。OLS を行なう行の範囲を変更するには SAMPLE 命令を使い始点と終点を指定します。なお、＄から始まる変数だけがほしい場合や、OLS のあとの付随するテスト結果だけが必要な場合には、？のマークを OLS などのコマンドの前につけて出力結果を抑制して表示させないことも可能です。ラグなどをともなって、すべてのデータではなくて前後をカットして OLS する必要のなる場合のために、OLS Y X2 / BEG=2 END=5 などとして、スラッシュのあとに始めの行と最後の行を指定することもできます。GAUSS では組込みコマンド OLS を使うか、自作した LSE などの procedure をプログラム後半に置いてそれを呼び出して CALL で呼び出して使うことになります。GAUSS では、OLS の変数の範囲は、あらかじめ行列操作で、例えば、x2=x2[2:5,.]; y=y[2:5,.];などとして整形したものを設定します。

スケラー計算と列計算と行列計算

SHAZAM	GAUSS
GEN1 N=1+2+3	n=1+2+3;
GENR YLG=LAG(Y)	ylag=lag1(y);
MATRIX Z=Y*X2'	z=y*x2';

SHAZAM では列シリーズごとの計算と行列計算は、GENR と MATRIX のどちらかを計算式の前につけて区別をする。MATRIX がついた計算結果には SAMPLE 命令は適用されない。一方、上の例で GENR で新しく作成された Y のシリーズのラグ LAG(Y)は２行目からデータとしては有効なので、OLS などをする場合には SAMPLE 2 29 などと、２行目から始めるようにする。なお、スケラーまたは定数計算には GEN１を式の前につける。これらは計算ごとに型宣言をしていると考えてもよいだろう。一方、GAUSS ではすべての計算はスケラーであろうとベクトルであろうと行列計算である。区別はありません。

配列表記とワイルドカード（行列のケース）

SHAZAM	GAUSS
X(2,1)	x[2,1]
X(0,2)	x[.,2]

配列表記については、SHAZAM では丸括弧、GAUSS では四角括弧の中に書くという違い

があります。SHAZAM ではワイルドカード機能に０を割り当てています。GAUSS ではドットマークがワイルドカードに相当して、その行または列すべてという意味になります。

ワイルドカード（変数列のケース）

SHAZAM	GAUSS
X:2	x[.,2]

SHAZAM では行列の計算と変数ごとの列の計算を特に区別して考えている。上の変数のあとにコロン：がきている場合にはその後の数、ここでは２の X の２行目が、例えば、

GENR X2=X:2

などと X２に設定されて扱われる。１つ前の例の X(0,2)というのは行列演算だから、

MATRIX X2=X(0,2)

などと行列として扱わなくてはならない違いがある。しかしながら、GAUSS では、列であろうと何であろうと計算自体がすべて行列として扱われるので、そのような区別はない。

行列の整形マージ

SHAZAM	GAUSS
COPY X:1 X:3 X:4 A	a=x[.,1]~x[.,3]~x[.,4]
Z=X|Y	z=x~y

SHAZAM では DIM 命令や上で扱ったようなワイルドカード機能のほかに、特殊な行列の整形に関するコマンドがあります。それは COPY です。複数個の行数が等しい列または行列を最後の変数名、ここでは、A に水平方向にマージします。（なお、この COPY コマンドは、後述するプロシジャーのインプット変数の設定にも使われます。）これと同じようなことをするのに｜のマークがあります。SHAZAM の｜のマークは水平方向のマージです。

演算記号

SHAZAM	GAUSS
＋－＊／‘ @	＋－＊./‘ .*.

演算記号は、クロネッカー積は、SHAZAM では@、GAUSS では.*.となります。それ以外ははほぼ同じです。SHAZAM でも GAUSS と同様に、X'X というふうに＊のマークを省略して転置行列の場合には計算できます。

組込み関数

SHAZAM	GAUSS
LOG(X)	ln(x)
DIAG(X)	diag(x) or diagrv(x,v)
IDEN(3)	eye(3)
LAG(X,2)	lagn(x,2)
NOR(100,2)	rndn(100,2)
UNI(100,2)	rndu(100,2)

SHAZAM では LOG は自然対数のことである。GAUSS では log は常用対数のことであって、自然対数には ln を用いなくてはならない。SHAZAM では DIAG は対角行列を取り出して列に直すだけではなくて、その反対に列を零行列の上の対角行列にあてはめることにも使う。GAUSS では、前者には diag を後者には diagrv を使う。SHAZAM では単位行列を作るには IDEN を使う。ラグには LAG を使い、その第２要素にラグの次数を書く。なお第２要素のない場合、LAG(X)とすると１階のラグが作成される。一方、GAUSS では、１階のラグは lag1(x)、それ以外は lngn(x,3)などとする。SHAZAM では GAUSS の rndu や rndu に相当する正規分布乱数と一様分布乱数が作成できる。NOR と UNI というコマンドになる。一個の乱数の場合、NOR(1)とできる。その他、INV、DET、SQRT や EXP など基本的な行列操作の関数はほぼ同じである。行列計算は、SHAZAM では、MATRIX から始まる文で書く。列の概念の計算であれば、一部 GENR から始まる文でも計算できる。

Do ループ

SHAZAM	GAUSS
DO #=1,10	for i (1,10,1);
GEN1 I=#	
PRINT I	print i;
ENDO	endofor;

SHAZAM ではほかの言語と違って、ループで回すインデックスには＃などの記号を使う。そのインデックスのシークエンスには、イコールサインのあと数字をカンマで区切ってならべる。上の場合、＃の中味が１から 10 まで（１ステップ）で増加する。なお、GAUSS では上のように FOR ループで書いてもよいが一般的には do while または do until で書く。

```
i=1;
do while i<=10;
      print i;
i=i+1;
endo;
```

などと書けばよい。SHAZAM では、＃のほかに、％＄！？などの記号も２重ループよりも大きいループになった時に用いることもある。

ループからの離脱

SHAZAM	GAUSS
DO #=1,1000	for i (1,1000,1);
. . . . .	. . . . .
ENDIF(G.LT.0.0001)	if g>0.0001; break; endif;
ENDO	endfor;

SHAZAM では DO ループから抜け出るのに ENDIF 文を用いて、その丸括弧の中に論理条件を書く。論理条件には GAUSS の＞,＞＝,＜,＜＝,＝＝,／＝の意味に、SHAZAM では、ドットを前後に書いて.GT.,.GE.,.LT.,.LE.,.EQ.,.NE.などとする。丸括弧内が真である場合には、GAUSS でいうところの break がなされる。

IF 文

SHAZAM	GAUSS
列の場合	
IF(X.GT.2)	if x>2;
. . . . .	. . . . .
	endif;
スケラーの場合	

```
IF1(A.LT.2)                    if a<2;
. . . . . .                    . . . . . .
                               endif;
```

==========

SHAZAM では通常 IF 文は、列の操作を念頭にされている。また普通 GAUSS の endif;文に相当する IF クローズの終了を意味するものはつけない。スケラーでの IF 文は、SHAZAM では IF1 という文を使い区別する。IF 文も IF１文も、SHAZAM では、論理式は丸括弧の中に書かなければならない。GAUSS ではいかなる場合にでも endif;文は if の数だけ必要なことに注意されたい。

プロシジャー

==========

SHAZAM	GAUSS
PROC MOMENT	proc gmoment(x);
COPY [INDEP] X（または MATRIX X=[INDEP] ）	
	local m;
MATRIX M=X'X	m=x'x;
PRINT M	print m;
	retp(m);
PROCEND	endp;
READ Y /ROWS =3 COLS=3	let x[3,3]=
1 2 3	1 2 3
4 5 6	4 5 6
7 8 9	7 8 9
	;
INDEP:Y	
EXEC MOMENT	call gmoment(x);
STOP	

==========

SHAZAM では GAUSS の proc に相当するものは、「PROC　関数名」として中味に何かを書いて「PROCEND」で終わります。明確なインプット変数は通常はありませんが、その代わりに、形式的な変数である[　]に囲まれた任意の変数を使い外部から入力をします。上の場合、MATRIX X=[変数名]でも COPY [変数名] X でも同じ結果になります。PROC の内

部で何か計算と命令を行なわせます。ここでは X'X を計算して内部で画面表示させています。GAUSS ではローカル宣言を変数にしないといけません。必須です。また、リターンがある場合には retp( )の中にリターンの変数を書いてやります。最後は endp;となります。呼び出し方は、SHAZAM では、「EXEC 関数名」としますが、インプットの設定には、文字列の配列であるコロン：を使って、前の変数に後ろの変数を入れてやります。（通常の文字配列の設定では、X：INDEP1 INDEP2 INDEP3 などと複数の変数を設定します。）一方、GAUSS にはこういう設定の仕方は普通は行ないません。上の例では proc を前置していますが、普通は慣習上、後置します。リターンをとめる場合には call 文で呼び出します。なお、SHAZAM のプログラムの最後には、STOP 文が必ずどんなときでも必要です。GAUSS では、end;文はオプショナルで、なくてもかまいません。