TEMA 5
PROCEDIMIENTOS AVANZADOS Y LENGUAJE MACRO

1. PROCEDIMIENTO IML

1.1 INTRODUCCIÓN AL PROCEDIMIENTO IML

IML (**I**nteractive **M**atrix **L**anguage) es una herramienta bastante poderosa para programar en condiciones dinámicas e interactivas. Se pueden invertir matrices, obtener la generación de autovectores y autovalores, se pueden crear funciones y subrutinas propias utilizando módulos del procedimiento. Se accede a gran cantidad de subrutinas diseñadas para facilitar la programación y hacerla más eficiente. Como es una parte del Sistema SAS se puede acceder a conjuntos de datos SAS o a ficheros externos. Se pueden leer, crear y modificar conjuntos de datos SAS sin utilizar ningún paso DATA, sólo a través del procedimiento. Por ejemplo:

```
read all var{camisas} where(nombre=:"Smith");
```

crea el vector camisas con aquellos observaciones cuya variable nombre='Smith' utilizando los valores de la variable camisas.

IML tiene un conjunto completo de sentencias tales como DO/END, START/FINISH, DO iterativos, IF-THEN/ELSE, GOTO, LINK, PAUSE, y STOP, necesarios para controlar la programación.

Para comenzar a programar en el lenguaje IML de Sas se necesita la sentencia:

Proc Iml;

a continuación aparecería el código donde se realizen las operaciones pertinentes. Para salir del lenguaje IML, se necesita la sentencia:

quit;

1.2. COMO GENERAR MATRICES CON EL PROC IML

Una matriz se define a través de un nombre SAS. Su dimensión se encuentra definida por el número de filas y de columnas. Como siempre en SAS el punto asignado a un elemento de la matriz indica valor missing. Cuando aparecen números entre corchetes, representan factores repetidos. Las matrices pueden ser numéricas o carácter.

1.2.1 CREACION DE MATRICES LITERALES.

Al definir la matriz de forma literal, los elementos se encuentran entre llaves. Se utilizan comas para separar las filas de la matriz. Por ejemplo:

x={bobo tonto} crea la matriz x=(BOBO TONTO) carácter (si los valores carácter se escriben entre comillas se diferencian las minúsculas de las mayúsculas como no hemos escrito comillas las letras SAS las considera mayúsculas)

Una matriz que inicialmente es asignada como numérica puede convertirse en carácter o viceversa sin más que redefinirla.

1.2.2 CREACION DE MATRICES POR ASIGNACION DE OPERADORES COMUNES

Existen funciones escalares como el log o sqrt que pueden ser utilizadas para crear o manipular todos los elementos de la matriz. Además existen funciones específicas matriciales como INV o RANK, que operan en el conjunto de la matriz.
y=inv(x) crea la matriz y , inversa de la matriz x, siempre que se pueda (x debe tener el mismo número de filas que de columnas).

Veamos los operadores mas comunes entre matrices:

a+b suma cada uno de los elementos de estas dos matrices (que tienen que tener la misma dimensión). A partir de la versión 9.2 puede sumarse un vector a una matriz (de tal forma que si a es una fila, esa fila se suma a todas las de la matriz b.

a+1 suma a todos los elementos de la matriz el escalar 1.

> < = <= >= ^= estos 6 operadores sirven para comparar los elementos de 2 matrices. El resultado es otra matriz que contiene 0 y 1 si la comparación es falsa o verdadera respectivamente:
Ejemplo: a={1 3 5, 7 9 10} b={4 4 7, 1 5 12} entonces c= a>b crearía la matriz:

$$c=\begin{pmatrix}0 & 0 & 0\\ 1 & 1 & 0\end{pmatrix}$$

La comparación también se puede llevar a cabo con un escalar. c= a>3 produciría:

$$c=\begin{pmatrix}0 & 0 & 1\\ 1 & 1 & 1\end{pmatrix}$$

Cuando se llevan a cabo comparaciones condicionales, y a partir del resultado se desarrollan una nueva serie de sentencias, la comparación debe ser cierta para todos los elementos de la matriz.
If a > b then z=1; z valdrá 1 sólo si todos los elementos de a son mayores que los de b.

|| El operador concatenación, produce una nueva matriz formada por la unión horizontal de las matrices involucradas en la operación. Es decir se duplica el número de columnas.
Ejemplo: a={1 3 5, 7 9 10} b={4 4 7, 1 5 12} entonces c=a || b

$$c=\begin{pmatrix}1 & 3 & 5 & 4 & 4 & 7\\ 7 & 9 & 10 & 1 & 5 & 12\end{pmatrix}$$

/ divisor, divide cada elemento de la primera matriz entre el elemento correspondiente de la segunda matriz. También se puede utilizar un escalar como divisor o como dividendo. Si el divisor es cero el resultado que se obtiene es missing (.)

<> Operador máximo, selecciona el mayor valor del elemento correspondiente a dos matrices:

```
a={1 3, 2 5};  b={0 5, 1 4};  c = a<>b;
```
$c=\begin{pmatrix}1 & 5\\ 2 & 5\end{pmatrix}$

>< Operador mínimo. Idéntico al anterior pero en este caso devuelve el mínimo de los valores que se comparan.

: valor 1 : valor 2. Crea un vector fila (con valor2-valor1+1 términos) cuyo primer elemento el valor 1, el segundo es el valor 1 +1 y así sucesivamente hasta que alcanza el valor 2.

Ejemplo x=5:9; x=(5 6 7 8 9)

Si el valor2 es menor que el valor 1 en lugar de incrementarse de 1 en 1 lo que se produce es un decremento de 1 en 1.

Ejemplo: x=5:3; x=(5 4 3)

& Operador lógico (equivalente a And, que también se puede utilizar), compara dos matrices elemento a elemento para producir una nueva matriz. El elemento de la nueva matriz será 1 si ambos son distintos de cero, si alguno es igual a 0 el valor del elemento será 0.

| Operador lógico (equivalente a OR), igual que el anterior pero ahora sólo necesita que uno de los dos elementos sea distinto de 0 para que valga 1.

^ Operado lógico Not. Si el valor del elemento de la matriz es 0, el nuevo valor será 1

Ejemplo: a={1 3, 0 1}; z= ^a; z= $\begin{pmatrix}0 & 0\\ 1 & 0\end{pmatrix}$

* Multiplicación de matrices. (Multiplicación habitual de matrices, donde el número de columnas de la primera matriz debe coincidir con el número de filas de la segunda).

Multiplica los elementos de dos matrices (no las matrices).

Ejemplo: `a={1 3, 2 5};  b={0 5, 1 4};  c=a#b ;` $c=\begin{pmatrix}0 & 15\\ 2 & 20\end{pmatrix}$

Si se utiliza un escalar en lugar de una matriz, se obtiene el producto del escalar por todos los elementos de la otra matriz.

Se pueden multiplicar matrices con diferentes dimensiones:

Ejemplo `a={1 3, 2 5};  b={4, 10};  c=a#b ;` $c=\begin{pmatrix}4 & 12\\ 20 & 50\end{pmatrix}$

Elevación de todos los elementos de la matriz a la potencia que se indique.

```
a={1 3, 2 5};  nueva=a##2;
```
produce como resultado $nueva=\begin{pmatrix}1 & 9\\ 4 & 25\end{pmatrix}$

** Elevación de la matriz completa a la potencia que se señale. Ejemplo:

Ejemplo: *x={1 2 3, 4 5 6, 7 8 9}; y=1:3;* *c=x[2,y]*; selecciona la segunda fila y las columnas 1, 2 y 3. c=(4 5 6)

- Operador sustracción. Los elementos de la nueva matriz se obtienen como la resta de los elementos de la segunda matriz a los elementos de la primera.

` La comilla descendente, es el operador de la matriz traspuesta. B=A`; B será la matriz cuyas filas serán las columnas de A, y cuyas columnas serán las filas de A.

```
a={1 3, 2 5};  nueva=a';
```
resultado $nueva = \begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix}$

1.2.3.-CREACION DE MATRICES POR ASIGNACION DE SENTENCIAS-FUNCIONES Y SUBRUTINAS

Operaciones matriciales simples.

ABS (matriz); Función escalar que devuelve otra matriz con el valor absoluto de todos los elementos de la matriz origen.

CUSUM Calcula una matriz con la suma acumulada, es decir va acumulando los elementos de la matriz primero por filas y luego por columnas:

Ejemplo: a={1 2 ,5 –2}; b=cusum(a); el resultado es: b=$\begin{pmatrix} 1 & 3 \\ 8 & 6 \end{pmatrix}$

DET(matriz) Devuelve el valor del determinante de la matriz.
a={1 2 ,5 –2}; b=det(a); produce el resultado b=-12

DIAG (vector) Devuelve una matriz diagonal, con los valores del vector.

a={1 3 5}; *diagonal=diag(a);* resulta: $diagonal = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{pmatrix}$

EIGVAL(matriz) Devuelve un vector columna conteniendo a los autovalores (la primera columna del resultado igual que antes. Si el autovalor es real puro la segunda columna sobra –vale 0-)

a={1 2 ,5 –2}; `valores=eigval(a);`

EIGVEC(matriz) Devuelve una matriz conteniendo a los autovectores ortonormales. Cada columna de esta nueva matriz es un autovector.

a={1 2 ,5 –2}; `vectores=eigvec(a);`

EXP (matriz) Calcula la exponencial de una matriz:

d=exp(A); $d = e^A$

INT (matriz) Devuelve una matriz formada por los valores enteros de la matriz origen.

$\begin{pmatrix} 3 & 1 \end{pmatrix}$

a1={1 2 ,5 –2}; a2={0 6, 3 –1}; c=max(a1,a2); resulta $nueva = \begin{pmatrix} 1 & 6 \\ 5 & -1 \end{pmatrix}$

MIN (mat1, ...,matn) Devuelve el mínimo valor de cada posición de las matrices incluidas en el argumento. Puede ser numérico o alfanumérico.

a1={1 2 ,5 –2}; a2={0 6, 3 –1}; c=min(a1,a2); resulta $nueva = \begin{pmatrix} 0 & 2 \\ 3 & -2 \end{pmatrix}$

NCOL (matriz) Devuelve un escalar conteniendo el número de columnas de la matriz.
a={1 3 5, 2 1 0}; colum=ncol(a); resulta colum=3

NROW (matriz) Devuelve un escalar conteniendo el número de filas de la matriz.
a={1 3 5, 2 1 0}; fila=nrow(a); resulta fila=1

RANK Devuelve una matriz conteniendo los rangos de los elementos de la matriz. Cuando hay empate entre valores los rangos son asignados de forma arbitraria.
y={1 -1 3 2}; rangos=rank(y) ; resulta y={2 1 4 3}
y={1 -1 3 2 1}; rangos=rank(y) ; resulta y={2 1 5 4 3}

RANKTIE Devuelve una matriz conteniendo los rangos de los elementos de la matriz origen. Cuando hay empates se les otorga el rango medio.
y={1 -1 3 2 1}; rangos=ranktie(y) ; resulta y={2.5 1 5 4 2.5}

SQRT Esta función calcula la raíz cuadrada de todos los elementos de una matriz, devolviendo una matriz de las mismas dimensiones que la original.

a={1 9, 4 16}; raiz=sqrt(a); resulta $a = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$

SSQ (mat1,mat2,...) Devuelve un escalar que es la suma de cuadrados no corregida de todos los elementos contenidos en las matrices indicadas:
y1={1 -2 3 2 1}; y2={0 2 -2 4}; suma_cua=ssq(y1, y2); resulta suma_cua=43

SUM Esta función devuelve un escalar con la suma de todos los elementos de la matriz o matrices incluidas como argumentos. La sintaxis es: SUM(mat1,...,matn);
y1={1 -2 3 2 1}; suma=sum(y1); el resultado es por tanto: suma=5;

T Esta función devuelve la traspuesta de la matriz que se incluya en el argumento. La sintaxis es : T(matriz)

a={1 5, -1 0}; traspuesta=t(a); resulta $traspuesta = \begin{pmatrix} 1 & -1 \\ 5 & 0 \end{pmatrix}$.

TRACE Esta función suma los elementos de la diagonal de la matriz incluida como argumento.
a={1 9, 4 16}; traza=trace(a); resulta traza=17.

VECDIAG Esta función crea un vector columna conteniendo los elementos de la diagonal de los argumentos incluidos. La sintaxis es: VECDIAG(matriz cuadrada).

Función BLOCK. Crea una matriz diagonal por bloques, en la que cada matriz origen ocupa un bloque. Los argumentos de la función representan los bloques de la matriz: La sintaxis es: BLOCK(mat1,mat2,..).
Ejemplo:

BLOCK(a,b,c,d) crea la matriz por cajas $\begin{pmatrix} (a) & & & \\ & (b) & & \\ & & (c) & \\ & & & (d) \end{pmatrix}$, ejemplo:

a={1 2, 3 4}; b={-1 -2 , 5 6}; c=block(a,b); c= $\begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 5 & 6 \end{pmatrix}$

Función J (n,m,w): Crea matrices cuyos elementos son todos iguales y coincidentes con el argumento determinado en la función, el primer valor (n) hace referencia a las filas, el segundo (m) a las columnas y el tercero 8w) al valor que le vamos a dar a todos los elementos de esa matriz
Ejemplo:

a=J(2,3,5) crea la matriz a= $\begin{pmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{pmatrix}$

Función I: Crea matrices identidad (hay que añadir la dimensión).

a= I(3) crea la matriz a= $\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$

DIAG(Matriz) Devuelve una matriz que simplemente es la diagonal de la matriz argumento, con los elementos que no pertenezcan a ella iguales a 0. Pero si el argumento es un vector, se crea una matriz diagonal cuyos elementos son los valores del vector.

DO (nºinic, nºfin, incremento) Crea un vector fila con los valores indicados en la sentencia
Ejemplo: i=do(2,18,4) ; produce como resultado i=(2 6 10 14 18).

1.3- SENTENCIAS PARA CREAR MATRICES A PARTIR DE CONJUNTOS DE DATOS Y VICEVERSA

USE c.datos sas Esta sentencia abre un conjunto de datos SAS (que será al que se refieran las siguientes sentencias que hagan llamadas a un conjunto de datos).

READ Lee variables del conjunto de datos SAS en matrices columna si se utiliza la opción VAR o en una única matriz si se utiliza la opción INTO. Cuando se utiliza esta última, cada variable en la opción VAR se convierte en una columna de la matriz objetivo y todas las variables de la opción VAR deben ser del mismo tipo. Si no se indica la opción VAR, las variables dentro de la opción INTO son numéricas.

CHAR,_NUM_, (todas, todas las alfanuméricas, todas las numéricas respectivamente.).ejemplos: var('tiempo1':'tiempo8'), var *_num_; etc*..
La opción WHERE selecciona observaciones dentro del rango especificado de acuerdo con la condición
Ejemplos:

```
read all where(alfa='b') into z;
```

Crea la matriz Z, con todas las observaciones en las que la variable alfa sea igual al valor b. Las columnas de la matriz son las variables numñericas del conjunto de datos que este en uso.

CREATE nombre — Crear un conjunto de datos. Sirve tanto para introducir como para obtener información. Existen dos formas de sintáxis:
CREATE conjunto de datos SAS <VAR operandos>; ó
CREATE conjunto de datos SAS from –nombre <colname= variable rowname=variable>
Además se requiere la sentencia APPEND para hacer que la matriz sea un nuevo conjunto de datos SAS. Si no se utilizara dicha sentencia el nuevo conjunto de datos no tendría observaciones.

APPEND [var operar] — Sirve para añadir datos al final del actual conjunto de datos que se crea con la sentencia CRETE.

CLOSE c.datos sas — Sirve para cerrar el conjunto de datos que se este creando y se especifique en la sentencia. Si no se indica ninguno se cierran todos los que se están creando.

1.4-FUNCIONES BASICAS.

PRINT — Imprime matrices. A veces se incluyen expresiones, puntos de control, mensajes y opciones para que solo se efectúe bajo esas condiciones:
También permite opciones como identificar nombres para las columnas y las filas de la matriz, colname = vector, o rowname = vector; e incluso cambiar el formato format= formato;
Ejemplo: print A format 4.2 ; (los valores sólo contienen dos cifras decimales).

QUIT — Salir de IML

RESET — Cambia las opciones por defecto que aparecen en las sentencias Print, show, etc.
La sintaxis es: RESET <opciones>;
Donde las opciones posibles son:

CENTER NOCENTER, salida centrada (por defecto CENTER)
DEFLIB= , librería por defecto (USER o WORK si no lo cambias)
DETAILS NODETAILS Solicita información adicional a cerca de las operaciones realizadas. Por defecto nodetails.
FUZZ NOFUZZ. Expresa que los números pequeños sean 0. Por defecto Nofuzz.
FW=número Manifiesta la anchura de la fuente. Por defecto vale 9.
LINESIZE=n Tamaño de línea (78 por defecto).
PAGESIZE=n Tamaño de la página (por defecto 21)

1.5 **CREACION DE SUBMATRICES, OPERANDO [ ]**

Los corchetes como operando sirven para bastantes operaciones en las que se necesita operar con submatrices de una matriz dada. Las mas relevantes son:

1. Selección de un elemento; x[2,3] , selecciona el elemento correspondiente a la segunda fila, tercera columna.
2. Selecciona una fila o columna. x[,3], selecciona la tercera columna entera. Mientras que x[2,] selecciona la segunda fila.
3. Selecciona submatrices:
   x[{1 3 }, {2 4}] selecciona las filas 1 y 3, y las columnas 2 y 4.
   x[1:4, 2:3] selecciona las filas 1,2,3 y 4 y las columnas 2 y 3.
4. Asignación de valores:
   x[2,3]=-1 asigna al elemento correspondiente a la fila segunda columna tercera el valor -1.
   x[, 4] ={0,0,-1,1} cambia la 4 columna de x y la convierte en : $\begin{pmatrix} 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}$
5. Reducción del número de filas y columnas debido a operaciones dentro de ellas, así se tiene que:
   A[+,] convierte la matriz A ($m \times n$) en un vector $1 \times n$, donde cada columna corresponde a la suma de todas las filas. (Es decir: el + correspondiente a las filas, y pide que se sume a través de todas las filas.
   A[+, < >] primero suma en todas las filas, y después busca el máximo en todas las columnas. De esta forma si A= $\begin{pmatrix} 1 & -1 & 2 \\ 7 & 2 & 3 \\ 3 & 5 & 4 \end{pmatrix}$, entonces A[+, < >] produce 11, al ser el máximo de (11,6,9).
   A [, < >] [+,] primero calcula el máximo para cada fila y a continuación las suma.
   En el ejemplo: el valor resultade es 14 que se obtiene como suma de maximos: $\begin{pmatrix} 2 \\ 7 \\ 5 \end{pmatrix}$
   ojo el operador <:> devuelve el índice del máximo, mientras que >:< devuelve el índice del mínimo. (con dos puntitos).
   A[,<:>] devuelve $\begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$, que son las columnas donde se encontraban los valores máximos de cada fila.
   A[:] devuelve la media de todos los elementos de la matriz A, es decir 2,888

FINISH;

.....

run mimodulo (x,y,z);

Devuelve las matrices X e Y conteniendo los senos y logarítmos de los elementos de la matriz z.

Las sentencias principales de programación de subrutinas son:

CALL nombre <argumento> llama a una función o subrutina , donde el nombre es un modulo definido a través de una subrutina IML o una función.
los argumentos son aquellos que pertenezcan al módulo o subrutina.

FINISH (módulo) Determina la terminación del módulo que se indique.

RUN Ejecuta las sentencias de un módulo. La sintaxis es RUN <nombre> <(argumentos>,

START y FINISH Estas sentencias definen a un módulo. La sintaxis es:
START <nombre> <argumentos> <GLOBAL(argumentos)>; sentencias del módulo; FINISH <nombre>;
Los argumentos son nombres de variables en el módulo (variables entrada o salida). los que aparezcan en la opción GLOBAL son tratados como variables globales, los demás son locales, relativos exclusivamente al módulo.

1.7 INTRODUCCION DE LA PROGRAMACIÓN R EN SAS

Se pueden traer funciones y programas (secuencia de sentencias) de R a SAS dentro del procedimiento IML. Para ello lo unico que hay que hacer es utilizar la sentencIa
SUBMIT / R;
a continuación escribir todas las sentencias en R necesarias para conseguir lo que se pretende
y al final de esas sentencias incluir la sentencia SAS

ENDSUBMIT;

Veamos un ejemplo, en verde se encuentra el codigo R que crea dos matrices y multiplica una por la traspuesta de la otra.

```
proc iml;
submit / R;
  rx <- matrix( 1:3, nrow=1)              # vector of sequence 1,2,3
  rm <- matrix( 1:9, nrow=3, byrow=TRUE)  # 3 x 3 matrix
  rq <- rm %*% t(rx)                      # matrix multiplication
  print(rq)
endsubmit;
```

En SAS la programación del sombreado en verde sería:

```
rx={1:3};
rm= {1 2 3, 4 5 6, 7 8 9);
 rq=rx*t(rm);
```

Por tanto una macro es un conjunto de setnecias macro y sentencias SAS, que se puede referenciar por otras partes del programa. Las macros permiten la inclusión de argumentos a los que denominamos macro-variables:

2.1- Creación de macros.

Para crear una macro realizamos las siguientes acciones,

1-Escribimos la sentencia:

%MACRO nombre de la macro (argumentos) ;

a continuación escribimos las sentencias que creemos oportunas y que formarán el cuerpo de la macro

por último escribiremos el final de la macro

%MEND;

Los argumentos de la macro son variables locales, que sólo tienen validez en esa macro. La referencia dentro de la macro a esos argumentos se realizará anteponiendo el símbolo ampersand (&) delante del nombre del argumento.

- Se define una macro variable como una variable que transfieren su valor a lo largo de varios pasos DATA y pasos PROC. Se diferencian las macro variables globales que tienen valor en un programa desde su creación y las locales que sólo tienen valor durante la ejecución de la macro (se crean justo en la macro). Una macro variable global se puede crear en cualquier parte de un programa con la sentencia:

%LET macro-variable (valor inicial);

por ejemplo esta sentencia crea la variable global nueva_n con valor inicial 5 (puede variar a lo largo de la programación):

```
%LET NUEVA_N (5) ;
```

Para llamar a una macro variable basta escribir el simbolo del procentaje % delante del nombre de la macro. Lógicamente esa macro ha debido crearse antes.

Veamos un ejemplo de MACRO, simple ;

```
%MACRO descriptiva (Data= , Out= , var=, by=, estadistico=mean);

proc sort data=&data;
                  by &by; run;

proc means data= &data noprint;
   var &var;
    by &by;
   output  out= &out  &estadistico = ;
  run;

proc univariate data=&data normal;
     var &var;
      histogram &var / normal;
run;

    proc print data=&out; run;
```

%MEND; en un programa se puede hacer la llamada, asi la sentencia sería:

2.3 SENTENCIAS MACRO COMUNES;

```
%DO  macro-variable= n %to m  %by  k;   /*bucle el valor de la variable varia de n a m de k en k valores*/
   sentencias;
%END;
```

Tambien se pueden utilizar

```
 %DO %WHILE
 %DO %UNTIL;
%IF    %THEN;

%ELSE;
```