SAS IML

Module de programmation

SAS : Statistical Analysis System
IML : Interactive Matrix Language

IML est un langage de programmation (hérité de l' APL et proche de MATLAB)
C'est un complément indispensable aux PROC de SAS, il se présente sous forme d'une PROC

PROC IML;

instructions

QUIT;

IML opère sur des éléments simples :

scalaire (numérique)

chaîne de caractères (caractère)

Mais aussi et surtout sur des matrices (de type numérique ou caractère)

Saisie d'une matrice

x = { ligne1, ligne2,…..,lignem } ; F3 pour soumettre (exécuter)

ligne = a1 a2 …. an (les touches Alt Gr 4 et Alt Gr + donnent respectivement { et })

Exemples

x = { 1 2, 4 5, 7 8 }; print (type(x)) , x ; suivi de F3

Renvoie dans la fenêtre OUTPUT (⟶)

N (x est une matrice de type numérique)

x

1 2

4 5

7 8

z = {'Paul' a , 'Pierre' b}; print z;

⟶

Paul A

Pierre B

w= { 'Paul' 18, 'Pierre' 14} ; x=type(w); print x; ⟶ C

Transposition d'une matrice

t(matrice); ou matrice`;  (la touche Alt Gr 7 donne `)

Exemple y=t(x); print y; → 1 4 7
2 5 8

Si z est une matrice de format (m, n)

nrow(z); → m et ncol(z); → n

Codage pour une valeur manquante

. Point pour une donnée de type numérique

‘ ‘ espace pour une donnée de type caractère

Concaténation horizontale || et verticale // de matrices (N ou C) || (2 fois Alt Gr 6)

Exemple a={1 7 , 6 2}; b={0 8, 4 1};

a||b → 1 7 0 8
6 2 4 1

a//b → 1 7
6 2
0 8
4 1

plus fin cf : insert(a,b,m,n) b insérée dans a devant la ligne m ou la colonne n

Opérations sur les matrices pouvant agir globalement ou terme à terme

Exemples d'opérations globales :

INV(x); $\longrightarrow$ x^{-1}

MIN(x); $\longrightarrow$ $Min(x_{ij})$

MAX(x); $\longrightarrow$ $Max(x_{ij})$

SUM(x); $\longrightarrow$ $\sum_{i,j} x_{ij}$

SSQ(x); $\longrightarrow$ $\sum_{i,j} x_{ij}^2$

DET(x); $\longrightarrow$ la valeur du déterminant de la matrice x

Exemples d'opérations terme à terme :

SQRT(x); $\longrightarrow$ $\left(\sqrt{x_{ij}}\right)$

LOG(x); $\longrightarrow$ $\left(\ln(x_{ij})\right)$

EXP(x); $\longrightarrow$ $\left(e^{x_{ij}}\right)$

Opérateurs unaire et binaires

Unaire -matrice; (matrice opposée)

Binaires + et - matrice1 + matrice2; matrice1 - matrice2;

matrice + scalaire; matrice - scalaire;

scalaire1 + scalaire2; scalaire1 - scalaire2;

Exemple

Si $a = \begin{bmatrix} 1 & 7 \\ 6 & 2 \end{bmatrix}$ alors $b = a - 2; \rightarrow \begin{bmatrix} -1 & 5 \\ 4 & 0 \end{bmatrix}$

Comparaisons < , > , = , <= , >= , ^= (Alt Gr 9 donne ^)

Comparaisons entre 2 matrices numériques de même format ou d'une matrice numérique et un scalaire. La comparaison se fait terme à terme et renvoie 1 si vrai et 0 si faux

Exemple : si a={1 7, 6 2}; b={0 8, 4 1};

$a > b; \rightarrow \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ b>=2; $\rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ b^=0; $\rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

Comparaisons pour le maximum <> et le minimum ><

matrice1 <> matrice2; ou matrice <> scalaire;

matrice1 >< matrice2; ou matrice >< scalaire;

La comparaison se fait terme à terme, on garde le terme qui satisfait le critère

$$Si\ a=\begin{bmatrix}1 & 7\\ 6 & 2\end{bmatrix}\ \text{et}\ b=\begin{bmatrix}0 & 8\\ 4 & 1\end{bmatrix}$$

$$a <> b;\ \rightarrow \begin{bmatrix}1 & 8\\ 6 & 2\end{bmatrix} \qquad a >< b;\ \rightarrow \begin{bmatrix}0 & 7\\ 4 & 1\end{bmatrix}$$

$$a <> 2;\ \rightarrow \begin{bmatrix}2 & 7\\ 6 & 2\end{bmatrix} \qquad a >< 2;\ \rightarrow \begin{bmatrix}1 & 2\\ 2 & 2\end{bmatrix}$$

La division / s'effectue terme à terme

matrice1 / matrice2; matrice / scalaire; $a / b;\ \rightarrow \begin{bmatrix}. & .875\\ 1.5 & 2\end{bmatrix}$

Attention à ne pas confondre avec l'inverse

a*inv(b); qui renvoie ab^{-1}

La multiplication # (Alt Gr 3) s'effectue terme à terme

matrice1#matrice2; ou matrice#scalaire;

matrice(n,m)#vecteur_ligne(1,m); ou matrice(n,m)#vecteur_colonne(n,1);

exemples

d={10 100}; /* vecteur ligne */ e={10, 100}; /* vecteur colonne */

$$Si\ a=\begin{bmatrix}1 & 7\\ 6 & 2\end{bmatrix}\ \text{et}\ b=\begin{bmatrix}0 & 8\\ 4 & 1\end{bmatrix} \qquad a\ \#\ b;\ \rightarrow\ \begin{bmatrix}0 & 56\\ 24 & 2\end{bmatrix}$$

Attention à ne pas confondre avec le produit matriciel $a*b;\ \rightarrow\ \begin{bmatrix}28 & 15\\ 8 & 50\end{bmatrix}$

$$a\#d;\ \rightarrow\ \begin{bmatrix}10 & 700\\ 60 & 200\end{bmatrix} \qquad a\#e;\ \rightarrow\ \begin{bmatrix}10 & 70\\ 600 & 200\end{bmatrix}$$

La puissance # # (2 fois Alt Gr 3) s'effectue terme à terme

matrice1##matrice2; ou matrice##scalaire;

exemples

$$Si\ a=\begin{bmatrix}1 & 7\\ 6 & 2\end{bmatrix}\ \text{et}\ b=\begin{bmatrix}0 & 8\\ 4 & 1\end{bmatrix} \qquad a\#\#b;\ \rightarrow \begin{bmatrix}1^0 & 7^8\\ 6^4 & 2^1\end{bmatrix}=\begin{bmatrix}1 & 5764801\\ 1296 & 2\end{bmatrix}$$

$$a\#\#0.5;\ \Leftrightarrow\ \text{sqrt(a);}\ \rightarrow \begin{bmatrix}\sqrt{1} & \sqrt{7}\\ \sqrt{6} & \sqrt{2}\end{bmatrix}=\begin{bmatrix}1 & 2.6457\\ 2.4494 & 1.4142\end{bmatrix}$$

$$a\#\#2;\ \rightarrow \begin{bmatrix}1^2 & 7^2\\ 6^2 & 2^2\end{bmatrix}=\begin{bmatrix}1 & 49\\ 36 & 4\end{bmatrix}$$

Attention à ne pas confondre avec les puissances de matrice

Matrice**scalaire; avec scalaire entier >=-1

a**(-1); ⇔ inv(a);

a**2; ⇔ a*a; renvoie $\begin{bmatrix}1 & 7\\ 6 & 2\end{bmatrix}\begin{bmatrix}1 & 7\\ 6 & 2\end{bmatrix}=\begin{bmatrix}4\,3 & 2\,1\\ 1\,8 & 4\,6\end{bmatrix}$

Index

valeur1 : valeur2; *renvoie toujours un vecteur ligne*

Si valeur1 <= valeur2 ⟶ valeur1 à valeur2 par pas de 1

2:6; ⟶ 2 3 4 5 6 (vecteur ligne)

Si valeur1 >= valeur2 ⟶ valeur1 à valeur2 par pas de -1

6:2; ⟶ 6 5 4 3 2 (vecteur ligne)

Progression arithmétique croissante et décroissante

do(début,fin,pas); *renvoie toujours un vecteur ligne*

Si début <= fin alors pas>0 do(3,10,2); ⟶ 3 5 7 9

Si début >= fin alors pas<0 do(10,3,-2); ⟶ 10 8 6 4

Indiçage et Sélection de sous matrices

matrice[restriction des lignes , restriction des colonnes]; ou matrice[restriction];

a={1 2 3,4 5 6, 7 8 9};

a[2,3]; ⟶ 6 a[9]; ⟶ 9

$$a\ [\ ,\ 3\ ];\ \rightarrow \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

$$a[1:2,\ ];\rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$a[2,2:3];\rightarrow \begin{bmatrix} 5 & 6 \end{bmatrix}$$

Opérateurs de **Réduction** des lignes ou des colonnes d'une matrice

+	réduction par addition
#	réduction par multiplication
:	réduction par la moyenne
##	réduction par la somme des carrés
<>	réduction par recherche du maximum
><	réduction par recherche du minimum
<:>	réduction par recherche de l'indice du maximum
>:<	réduction par recherche de l'indice du minimum

exemples

$$a = \begin{bmatrix} 1 & 2 & 0 \\ 5 & 4 & 3 \\ 8 & 7 & 6 \end{bmatrix}$$

min max

1 2 0 5 4 3 8 7 6

1 2 3 4 5 6 7 8 9

a[>:<]; ⟶ 3 a[<:>]; ⟶ 7 a[##]; ⇔ ssq(a); ⟶ $1^2+2^2+\ldots\ldots+6^2=204$

a[+]; ⇔ sum(a); ⟶ 1+2+….+6=36 a[:]; ⟶ 36/9=4

a[<>]; ⇔ max(a); ⟶ 8 a[><]; ⇔ min(a); ⟶ 0

a[+,]; ⟶ [14 13 9] réduction des lignes par la somme

$$a = \begin{bmatrix} 1 & 2 & 0 \\ 5 & 4 & 3 \\ 8 & 7 & 6 \end{bmatrix}$$

a[,+]; ⟶ $\begin{bmatrix} 3 \\ 12 \\ 21 \end{bmatrix}$ réduction des colonnes par la somme

a[<:>,]; ⟶ [3 3 3] réduction des lignes par la recherche de l' indice du maximum

a[,><]; ⟶ $\begin{bmatrix} 0 \\ 3 \\ 6 \end{bmatrix}$ réduction des colonnes par recherche du minimum

a[<>,]; ⟶ [8 7 6] réduction des lignes par recherche du maximum

a[,:]; ⟶ $\begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$ réduction des colonnes par la moyenne

a[+,][<>]; ⟶ [14 13 9] [<>] ⟶ 14 a[<>,][+]; ⟶ [8 7 6] [+] ⟶ 21

a[,><][:]; ⟶ $\begin{bmatrix} 0 \\ 3 \\ 6 \end{bmatrix}$ [:] ⟶ 3

Exercices :

Pour une matrice numérique donnée

a) Indiquer le numéro du vecteur colonne qui à la plus faible norme euclidienne.

b) Donner la valeur de la plus forte norme euclidienne des vecteurs colonnes.

Commandes de remplissage et de (re)formatage

repeat(matrice, nombre en ligne, nombre en colonne);

Exemple

x={1 2 0};

repeat(x,1,3); → [1 2 0 1 2 0 1 2 0]

repeat(x,3,1); →

$$\begin{bmatrix} 1 & 2 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \end{bmatrix}$$

repeat(x,3,2); →

$$\begin{bmatrix} 1 & 2 & 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 1 & 2 & 0 \end{bmatrix}$$

unique(matrice); → un vecteur *ligne* contenant les termes distincts et ordonnés

x={0 2 0,1 1 1,. 2 0} ; → . 0 1 2

Faire les exercices 1 et 2 de la feuille EXERCICES : IML

shape(matrice , nlig , ncol <,valeur>); ⟶ une matrice(nlig,ncol)

Exemples

$$x = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \Leftrightarrow 1\ 2\ 3\ 4\ 5\ 6$$

shape(x,2,3); ⟶ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

shape(x,2,2); ⟶ $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

shape(x,3,3); ⟶ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix}$

shape(x,3,3,0); ⟶ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 0 & 0 \end{bmatrix}$

J(nlig , ncol <,val>); val est N ou C par défaut val=1

J(2,3); ⇔ shape(1,2,3); ⇔ repeat(1,2,3); ⟶ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

I(n); ⟶ matrice Identité de R^n

I(3); ⟶ $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

diag(matrice); ⟶ matrice diagonale

Si $a=(a_{ij})$

diag(a); ⟶ $\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & O & 0 \\ 0 & 0 & a_{nn} \end{bmatrix}$

vecdiag(a); ⟶ $\begin{bmatrix} a_{11} \\ M \\ a_{nn} \end{bmatrix}$

Commande très utile dans IML

loc(matrice Numérique); renvoie dans un vecteur ligne les positions des éléments non nuls (vrais) de la matrice. Les données manquantes (.) sont traitées comme des zéros.

Vrai = valeur non nulle ou non manquante (^= .)

Faux = 0 ou manquant (.)

exemple $a = \begin{bmatrix} 1 & 0 \\ -2 & 3 \\ 0 & -1 \end{bmatrix} \Leftrightarrow \begin{cases} 1 & 0 & -2 & 3 & 0 & -1 \\ (1) & (2) & (3) & (4) & (5) & (6) \end{cases}$

loc(a); $\longrightarrow$ $[1 \quad 3 \quad 4 \quad 6]$

a>0; $\longrightarrow$ $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \Leftrightarrow \{1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0$

loc(a>0); $\longrightarrow$ $[1 \quad 4]$

a[loc(a>0)]; $\longrightarrow$ $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ On a extrait les termes >0 de a

<u>Exercices :</u>

Pour une matrice numérique x donnée.

a) *Ne garder que ces éléments >0 et mettre les autres éléments à 0.*

b) *Même chose mais en mettant les autres éléments à manquants.*

c) *Quelle différence y a-t-il entre sum(x^=.); et ncol(loc(x));*

Les structures de contrôle

IF condition THEN instruction1; < ELSE instruction2; >

Instruction DO; ---------- END;

Exemple

```
If  x>=0   then
        do;  y=sqrt(x);   print  y [format=7.2];   end;
    else
        print ‘’Erreur : valeurs négatives ‘’;
```

x={1 2, 4 -7}; ⟶ ’Erreur : valeurs négatives

Traitements Itératifs : les boucles DO

DO définition de la boucle ;

paquet d’instructions qui sera répété (itération)

END;

1) DO variable = début TO fin <BY incrément>; par défaut l'incrément vaut 1

do i=1 to 5; .... end; pour i=1,2,3,4,5 répéter l'itération

do j=-10 to 100 by 20; …. end;

pour j=-10,10,30,50,70,90 répéter l'itération

Exemple

Générer : {9 99 999 9999 99999} <=> {10-1 10^2-1 10^3-1 10^4-1 10^5-1}

```
b=J(1,5,10);  q=10;
do i=2 to 5;
    b[i]=q*b[i-1];
end;
b=b-1; print b;
```

2) Boucle tant que (WHILE) : DO WHILE(condition);

Tant que la condition est vraie (^=0 ou ^=.) répéter l'itération. On s'arrête dès que la condition est fausse (=0 ou .)

La condition étant évaluée en *début* d'itération, on n'est pas sûr de passer dans la boucle

Exemple

```
s=1;
do while(s<11);
      print s;
      s=s+2;
end;
```

→ 1 3 5 7 9

s	condition	output
1	V	1
3	V	3
5	V	5
7	V	7
9	V	9
11	F	

3) Boucle jusqu'à ce que (UNTIL) : DO UNTIL(condition);

Répéter jusqu'à ce que la condition soit vraie (^=0 ou ^=.) , on s'arrête dès que la condition est vraie.
La condition étant évaluée en *fin* d'itération, on est sûr de passer dans la boucle

Exemple

```
s=1;
do until(s>11);
    print s;
    s=s+2;
end;
```

→ 1 3 5 7 9 11

s	condition	output
1		1
3	F	3
5	F	5
7	F	7
9	F	9
11	F	11
13	V	

4) DO mixtes : 1) + 2) ou 1) + 3)

DO var = debut to fin <BY pas> WHILE(condition);

DO var = debut to fin <BY pas> UNTIL(condition);

NB: l'incrémentation est toujours faite avant d'évaluer la condition

Exemple

Dire si un entier tiré au hasard dans [0,10^5] est inférieur ou égal à 10 ,100,1000, .. , 10^5.

```
x= int(1e5*uniform(0));
borne=1; q=10;  ok=0;
do i=1 to 5  until(ok);
        borne=borne*q;
        if x <=borne then  do;  print '' x est <= à '' borne; ok=1;  end;
end;
```

Faire les exercices 3, 4, 5, 6, 7 et 8 de la feuille EXERCICES : IML

MODULES ou PROGRAMMES IML

- Fonction qui renvoie 1 résultat unique
- Procédure qui renvoie plusieurs (>=2) résultats

Syntaxe pour une procédure

START nom de la procédure (liste d'arguments);

| liste de commandes

FINISH nom de la procédure ;

Liste d'arguments : en tête de liste les arguments en sortie (les résultats)

en fin de liste les arguments en entrée (les données)

Appel d'une procédure

RUN nom de la procédure (liste d'arguments);

Syntaxe pour une fonction

START nom de la fonction (liste d'arguments en entrée);

```
-------
RETURN(resultat);
-------
```

FINISH nom de la fonction ;

Appel d'une fonction

result = nom de la fonction (liste d'arguments);

Exemples de fonctions :

Centrage des colonnes d'une matrice (Cf : centrage.sas)

Puis centrage ET réduction des colonnes d'une matrice (Cf: cered.sas)

Peut-on utiliser ces programmes tels quels avec des données manquantes ?

(Cf: zered.sas)

Centrage.sas version 0

```
start centrage(y);
n=nrow(y);
 yc=y-repeat(y[:,],n,1);
return(yc) ;
finish centrage;
```

Problème de la version 0

```
x=1:7;       /* x est un vecteur ligne */
xc=centrage(x);  print xc;
```

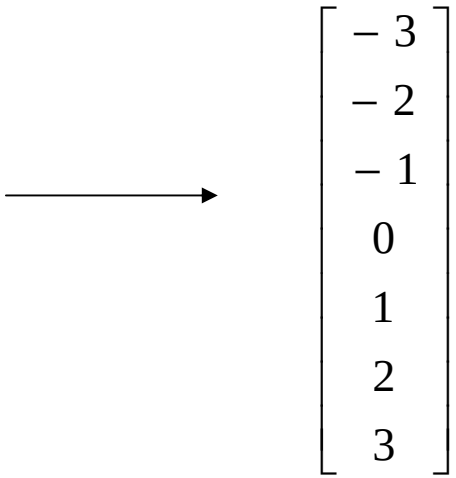
renvoie [0 0 0 0 0 0 0]

Centrage.sas version 1

```
start centrage(y);
n=nrow(y);
 if n=1 then  do; n=ncol(y); y=y`; end;
yc=y-repeat(y[:,],n,1);
return(yc) ;
finish centrage;
```

C'est mieux ! mais le format de sortie diffère de celui de l'entrée

```
x=1:7;  xc=centrage(x);  print xc;
```



Centrage.sas version 2 (correcte)

```
start centrage(y);
n=nrow(y);
if n=1 then  yc=y-y[:];
        else  yc=y-repeat(y[:,],n,1);
return(yc) ;
finish centrage;

x=1:7; xc=centrage(x); print x , xc;
y=(repeat(t(1:7),1,3)); yc=centrage(y); print y  yc;
```

$$x = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

$$xc = \begin{bmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \\ 5 & 5 & 5 \\ 6 & 6 & 6 \\ 7 & 7 & 7 \end{bmatrix}$$

$$yc = \begin{bmatrix} -3 & -3 & -3 \\ -2 & -2 & -2 \\ -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

/* cas des données manquantes */

z={. 2 3 4 5 6 7,1 2 3 . 5 6 7,. 2 3 4 5 6 ., ...}`;

$$z = \begin{bmatrix} . & 1 & . & . \\ 2 & 2 & 2 & . \\ 3 & 3 & 3 & . \\ 4 & . & 4 & . \\ 5 & 5 & 5 & . \\ 6 & 6 & 6 & . \\ 7 & 7 & . & . \end{bmatrix}$$

```
print z;
som=z[+,]; print som;
eff=j(1,4,0);
do k=1  to 4; eff[k]=sum(z[,k]^=.); end;
print eff;
mean= som/eff; print mean;
moy=z[:,]; print moy;
zc=centrage(z);
print zc;
```

$$som = \begin{bmatrix} 28-1=27 & 28-4=24 & 28-8=20 & 0 \end{bmatrix}$$

$$eff = \begin{bmatrix} 6 & 6 & 5 & 0 \end{bmatrix}$$

$$mean = \begin{bmatrix} 27/6=4.5 & 24/6=4 & 20/5=4 & . \end{bmatrix}$$

$$moy = \begin{bmatrix} 4.5 & 4 & 4 & . \end{bmatrix}$$

$$zc = \begin{bmatrix} . & -3 & . & . \\ -2.5 & -2 & -2 & . \\ -1.5 & -1 & -1 & . \\ -0.5 & . & 0 & . \\ 0.5 & 1 & 1 & . \\ 1.5 & 2 & 2 & . \\ 2.5 & 3 & . & . \end{bmatrix}$$

En conclusion centrage est utilisable avec des données manquantes

cered.sas *(centrer et réduire sans donnée manquante)*

```
start cered(y);
n=nrow(y); p=ncol(y);
if n=1 then
  do; ybar=y[:];
      yc=y-ybar;
      yect=sqrt(yc[##]/p);
      ycr=yc/yect;
  end;
       else
  do; ybar=y[:,];
      yc=y-repeat(ybar,n,1);
      yect=sqrt(yc[##,]/n);
      ycr=yc/repeat(yect,n,1);
  end;
return(ycr) ;
finish cered;

x=1:7; xcr=cered(x); print x, xcr;
y=(repeat(t(1:7),1,3)); ycr=cered(y); print y, ycr;
```

zered.sas *(centrer et réduire avec ou sans données manquantes)*

```
start zered(y);
n=nrow(y); p=ncol(y);
if n=1 then
 do; ybar=y[:]; yc=y-ybar;
    pp=sum(y^=.); yect=sqrt(yc[##]/pp);
    ycr=yc/yect;
 end;
       else
 do;
     eff=j(1,p,0);
     do j =1 to p; eff[j]=sum(y[,j]^=.); end;
     ybar=y[:,]; yc=y-repeat(ybar,n,1);
     yect=sqrt(yc[##,]/eff);
     ycr=yc/repeat(yect,n,1);
 end;
return(ycr) ;
finish zered;
```

Stockage et déstockage de programmes IML dans un catalogue IML permanent

Stockage

LIBNAME alias ‘chemin’;

< PROC IML; >

RESET STORAGE = alias.nom du catalogue;

STORE MODULE=nom du programme;

< SHOW STORAGE; > < QUIT; >

Exemple :

```
libname  a  ‘c:\cours\sas’;   proc iml;
reset storage=a.stage;            /* création et ouverture du catalogue IML */
store module=centrage;            /* stockage dans util de centrage */
show storage;                     /* visualisation du contenu d’util */
```

ceci implique la création du fichier c:\cours\sas\stage.sas7bcat

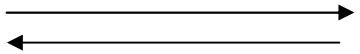
Déstockage pour utilisation *ultérieure* de programmes IML

```
libname  a  'c:\cours\sas';   /* définir le chemin d'accès */

proc iml;                     /* lancement d' IML */
reset storage=a.stage;        /* ouverture du catalogue IML */
show storage;                 /* visualisation du contenu d'util */
load module=centrage;         /* chargement du programme centrage */
< load module= _all_; >       /* chargement de tous les programmes du catalogue */

                  /* utilisation de centrage */
print (centrage(1:7));        ⟶   -3 -2 -1 0 1 2 3
<quit;>                                  /* sortie d'IML */
```

Créer et stocker sous forme de procédures les exercices 1 à 8 de la feuille EXERCICES : IML

Passage IML  Table SAS

1) préliminaire

```
/* création d'une table sas en ligne */
libname  a  'c:\cours\sas ';
data a.tab;
input Nom $ Age Taille Poids;
lines;
Paul 25 185 80
Pierre 23 170 90
Jacques 26 190 100
;
run;
proc contents data=a.tab position; run;
```

Nom	Age	Taille	Poids
Paul	25	185	80
Pierre	23	170	90
Jacques	26	190	100

'*'	Variable	Type	length
1	Nom	char	8
2	Age	num	8
3	Taille	num	8
4	Poids	num	8

2) Passage Table SAS vers IML

```
/* passage table sas --> matrices iml */
proc iml;
use a.tab;                              /* ouverture de la table sas */
read all var {'Nom'} into idobs;        /* lecture */
print idobs;                            /* création des matrices IML */
idvar={'Age' 'Taille' 'Poids'};
read all var idvar into x;
< read all var _num_ into x; >
print x;
close;                                  /* fermeture de la table sas */

show names;       /*
                  idobs (3,1)   char
                  idvar  (1,3)  char
                  x      (3,3)  num
                  */
print / "Matrice des Donnees";
print , x[rowname=idobs colname=idvar];
```

Matrice des Données

	x		
	Age	Taille	Poids
Paul	25	185	80
Pierre	23	170	90
Jacques	26	190	100

Syntaxe de USE et READ

USE nom de table <VAR sélection > <WHERE(condition)>;

READ <étendue> <VAR sélection > <WHERE(condition)> INTO nom de matrice;

sélection	liste de variables
	matrice contenant des noms de variables
	ALL toutes les variables
	NUM toutes les variables numériques
	CHAR toutes les variables caractères
étendue	ALL pour toutes les observations

3) Passage IML vers table SAS

/* première création d'une table sas à partir de vecteurs colonnes */

```
t=x[,2]; rang=t('r1':'r3');
create tab2 var {idobs t rang};
append; close;
```

/* dans le log : work.tab2 3 obs, 3 variables */

IDOBS	T	rang
Paul	185	r1
Pierre	170	r2
Jacques	190	r3

/* ajout de l'individu moyen */

```
y=x//x[:,];
ident=idobs//'Moyen';
print  ident y;
```

→

Paul	25	185	80
Pierre	23	170	90
Jacques	26	190	100
Moyen	24.6	181.6	90

/* deuxième création d'une table sas à partir d'une matrice*/

```
create a.tab3 from y [colname=idvar rowname=ident];
append from y [rowname=ident];
close; quit;
data a.tab3; set a.tab3; rename  ident=Nom;
proc print data=a.tab3; run;
proc contents data=a.tab3 position; run;
/* dans le log : a.tab3   4 obs, 4 variables */
```

Nom	Age	Taille	Poids
Paul	25	185	80
Pierre	23	170	90
Jacques	26	190	100
Moyen	24.6	181.6	90

Les 2 syntaxes de CREATE

1) à partir de vecteurs colonnes

CREATE nom de table VAR liste des noms de variables ; APPEND;   CLOSE;

2) à partir d'une matrice

CREATE nom de table FROM

matrice[COLNAME = identifiant des noms des variables

ROWNAME = identifiant des observations] ;

APPEND FROM   matrice[ROWNAME = même identifiant des observations] ;

CLOSE;