

Les macro-fonctions expliquées à ma fille

Beaucoup de programmeurs SAS connaissent les macro-variables, pratiques pour véhiculer de l'information (des paramètres) d'un point à un autre d'un programme. Egalement populaires, les macro-programmes, qui permettent d'encapsuler du code et de s'en resservir aussi souvent que nécessaire en changeant seulement quelques paramètres.

Mais les macro-fonctions ? A quoi servent-elles ? Peuvent-elles m'être utiles ?

Quelle différence avec une fonction SAS ?

Il y en a principalement deux :

1. une fonction SAS s'applique à la valeur d'une variable, stockée dans une table ; alors que la macro-fonction s'applique à la valeur d'une macro-variable, stockée en mémoire pour la durée de la session ;
2. une fonction SAS ne peut s'utiliser qu'à l'intérieur d'une étape Data, et plus occasionnellement à l'intérieur d'une procédure (SQL, IML, clauses WHERE) ; une macro-fonction peut être utilisée n'importe quand.

Le résultat d'une macro-fonction est toujours du texte : il s'agit en effet du seul type de données reconnu par le compilateur macro.

La gestion des opérations de base

Comme le compilateur macro ne gère que du texte, une opération comme $2+2$ sera considérée par défaut comme du texte. La macro-fonction %EVAL permet de forcer le compilateur macro à effectuer les quatre opérations de base (addition, soustraction, multiplication et division).

```
%PUT 2+2 ; /* affiche le texte 2+2 */
%PUT %EVAL(2+2) ; /* affiche le texte 4 */

/* incrémentation d'une macro-variable : */
%LET i = %EVAL(&i + 1) ;
```

Les limites de %EVAL sont qu'il ne sait traiter que les opérations dans le monde des entiers (relatifs). Ainsi, %EVAL(5/2) renvoie le texte 2. Et %EVAL(2.5*2) renvoie une erreur (« Opérande caractère trouvé dans la fonction %EVAL ou condition %IF là où un opérande numérique est requis. La condition était : $2.5*2$ ») à cause du séparateur décimal, que %EVAL ne sait pas traiter.

Pour le cas où une opération sur des nombres décimaux ou avec un résultat décimal vous serait nécessaire (ce qui est rare dans les utilisations habituelles du macro-langage), on peut utiliser la macro-fonction %SYSEVALF à la place de %EVAL. %SYSEVALF connaît le monde merveilleux des nombres réels et sait additionner, soustraire, multiplier et diviser les nombres à virgule, pour un résultat décimal.

La gestion des chaînes de caractères

Il existe quelques macro-fonctions similaires aux fonctions SAS de traitement des chaînes de caractères ; le seul aménagement dans la syntaxe est l'absence de guillemets autour des

chaînes considérées (puisque pour le macro-langage, tout est du texte : pas besoin de guillemets pour les différencier du reste).

Ces macro-fonctions sont :

- %SUBSTR pour l'extraction d'une partie de la valeur d'une macro-variable
- %LENGTH pour la longueur d'une valeur de macro-variable
- %UPCASE et %LOWCASE pour la mise en majuscules / minuscules d'une valeur de macro-variable
- %SCAN pour l'extraction d'une partie d'une valeur de macro-variable en fonction de délimiteurs
- %INDEX pour la position d'un caractère dans la valeur d'une macro-variable
- %LEFT et %TRIM pour l'élimination de blancs à gauche ou à droite de la valeur d'une macro-variable
- %CMPRES pour l'élimination de blancs à gauche et à droite, et des blancs multiples (à la manière de la fonction COMPBL).

```
PROC SQL NOPRINT ;
  SELECT COUNT(*) INTO : nb
  FROM sashelp.class ;
QUIT ;
%PUT ***&nb ;
***      19
%PUT ***%LEFT(&nb &nb) ;
***19      19
%PUT ***%TRIM(&nb) %LEFT(&nb) ;
***19 19
%PUT ***%CMPRES( &nb &nb ) ;
***19 19
%PUT %SCAN(&sysscp) ;
WIN
```

La macro-fonction %SYSFUNC

Comme l'arsenal des macro-fonctions est relativement limité, surtout en comparaison du vaste ensemble de fonctions SAS disponibles, il était très utile de disposer d'une connexion entre le monde macro et les fonctions SAS. Il s'agit de « l'adaptateur » %SYSFUNC, à l'intérieur duquel on peut spécifier une (et une seule) fonction SAS, appliquée à la valeur d'une macro variable.

Un second argument optionnel permet d'appliquer un format au résultat renvoyé par la fonction SAS.

```
%PUT %SYSFUNC(TODAY()) ;
17135
%PUT %SYSFUNC(TODAY(), FRADFWKX.) ;
Jeudi 30 novembre 2006
%LET table = sashelp.class ;
%PUT %SYSFUNC(ATTRN(%SYSFUNC(OPEN(&table)),NOBS)) obs ;
19 obs
%PUT %SYSFUNC(TODAY(), FRADFWKX.) ;
Jeudi 30 novembre 2006
```

Utilitaires : %SYMEXIST, %SYMDEL, %SYSPROD, %SYSGET

Quelques macro-fonctions sont sans équivalent dans le monde des fonctions SAS :

- %SYMEXIST permet de valider l'existence d'une macro-variable (renvoie 1 si la macro-variable existe, 0 sinon)
- %SYMDEL permet de supprimer une macro-variable (cela libère un peu de mémoire vive si la macro-variable n'est plus utilisée)
- %SYSPROD avec comme argument un nom de module SAS, renvoie 1 si le module est sous licence, et 0 sinon
- %SYSGET permet de récupérer dans SAS la valeur d'une variable système (variables d'environnement).

```
%PUT %SYMEXIST(table) ;
1
%SYMDEL table ;
%PUT %SYMEXIST(table) ;
0
%PUT SAS/STAT : %SYSPROD(STAT) - SAS/QC : %SYSPROD(QC) ;
SAS/STAT : 1 - SAS/QC : 0
%PUT %SYSGET(processor_identifieur) ;
x86 Family 15 Model 3 Stepping 4, GenuineIntel
```

Le macro-quoting avec %STR

Certains caractères peuvent ponctuellement poser des problèmes dans un macro-programme. Ainsi, l'inclusion d'un point-virgule dans la valeur d'une macro-variable définie par un %LET : il y a confusion entre le point-virgule qui marque la fin de la valeur dans la syntaxe de %LET, et le point-virgule qui fait partie de la valeur à stocker.

De même, l'affichage d'une apostrophe dans un texte %PUT est source d'ennuis puisque SAS comprend par défaut cette apostrophe comme un guillemet ouvert (et jamais fermé, ce qui cause quelques désagréments dans l'exécution du reste des programmes).

Pour « protéger » ces caractères de l'incompréhension des compilateurs, on les cache à l'aide de fonctions dites de quoting. Il en existe une demi-douzaine (%NRSTR, %NRBQUOTE, %BQUOTE, etc.) mais la plus universelle (et bien souvent largement suffisante) est %STR. Elle s'utilise ainsi :

- %STR encadre un caractère ponctuellement piégeux s'il ne fonctionne pas habituellement par paires : point-virgule, virgule, signe égal, esperluette (&), etc.
- %STR encadre une apostrophe, un guillemet seul, une parenthèse ouvrante ou fermante seule, en la faisant précéder d'un signe %.

```
%PUT nombre d%STR('%')observations : &nb ;
nombre d'observations : 19
%LET listing = PROC PRINT %STR(;) RUN %STR(;) ;
```