Tutoriel 4: Macro Variables et Macro Commandes en SAS

Résumé

Cette vignette décrit brièvement les principes et objets du macro langage de SAS permettant d'écrire des macros commandes : macros variables, macros fonctions, passages de paramètres et syntaxe d'une macro commande. Ces éléments concernent le module SAS de base.

Plan des tuteuriels :

- *Prise en main*
- *Gestion des données*
- *Graphiques*
- *Macros-commandes*
- *Bases de données*

Les procédures du module SAS/STAT sont étudiées dans les cours de statistique afférents.

1 Introduction

1.1 Objectif

Le principal objectif de cette vignette est de s'initier à la réalisation de macros commandes SAS. Il s'agit de concevoir puis carrosser un ensemble de traitement spécifiques afin, par exemple, de les rendre accessibles à des utilisateurs non spécialistes de SAS mais gros consommateurs comme dans l'industrie pharmaceutique ou le marketing.

La mise au point d'une macro commande est délicate car les messages concernant des erreurs à l'intérieur d'une macro ne sont pas du tout explicites. Il est donc vivement conseillé, lorsque cela est possible, de suivre la démarche suivante :

1. Écrire et tester le programme qui exécute l'algorithme voulu.
2. Déterminer les variables qui seront les paramètres de la macro, celles qui rendront facilement généralisable à d'autres usages le programme ainsi conçu.
3. Transformer en macros variables les variables ainsi identifiées. Elles seront regroupées en tête de programme, initialisées et documentées.
4. Tester les fonctionnalités du programme en variant les valeurs des paramètres / macros variables.
5. Encapsuler le programme dans une macro commande (
6. Faire exécuter la macro commande à SAS.
7. Tester l'appel de la macro en respectant scrupuleusement l'ordre des paramètres.

Cette démarche élémentaire n'est pas réalisable pour des macros compliqués faisant intervenir des boucles. Il est quand même conseillé de progresser par étape pour éviter des talonnements chronophages à cause de messages d'erreur incompréhensibles.

1.2 Principes

Le principe général consiste à associer une *chaîne de caractères*, une suite de commandes ou, un texte à un *identificateur*. Par la suite, toute occurrence de cet identificateur ou macro variable est remplacée par le texte désigné au cours d'un traitement préalable à l'exécution proprement dite des commandes.

Le pré-processeur implicitement invoqué reconnaît différents objets : variables, commentaires, commandes, fonctions, arguments, qui lui sont propres (précédés des caractères `&` ou `%`) ; ils lui confèrent les possibilités d'un langage de programmation rudimentaire mais structuré.

Le macro-langage, au même titre, augmente les possibilités du langage de base. Il permet de passer des paramètres entre les étapes DATA et PROC et de systématiser l'enchaînement d'une séquence donnée d'instructions.

Les macro-variables et macro-commandes sont connues, sauf déclaration explicite contraire (`%global, %local`), dans l'environnement dans lequel

elles sont déclarées : globalement pour toute une session SAS ou localement à l'intérieur d'une macro.

2 Objets manipulés

2.1 Macro-variables

syntaxe

La déclaration d'une macro-variable consiste à associer par la commande `%let` une chaîne de caractères (jusqu'à 65534) à un identificateur (de 1 à 32 caractères) :

```
%let nomvar1=taille;
%let nomvar2=poids;
%let varlist=csp sexe age taille poids revenu;
/* affichage du contenu : */
%put &varlist;
```

Certaines déclarations de macros variables sont implicites : compteur d'une boucle `%do`, paramètres d'une macro-commande,....

Dans la suite du programme, les références à une macro-variable sont précédées du caractère `&` :

```
title "Étude des variables &varlist";
proc plot;
     plot &nomvar1*&nomvar2;
run;
```

Le caractère `&` est traduit au niveau du pré-traitement : remplacer la macro-variable qui suit par la chaîne de caractères avant de passer à l'exécution.

Attention aux chaînes de caractères. Si la chaîne est entre "... ", une macro variable contenue dans la chaîne est interprétée, remplacée par sa "valeur". Ce n'est pas le cas si la chaîne est entre '...'.

Une macro-variable peut contenir elle-même des commandes ou instructions SAS mais, dans ce cas, il est préférable de définir une macro-commande.

Le système gère des macros variables prédéfinies comme `sysdate` et `sysday` qui contiennent respectivement la date et le jour du début de la session SAS en cours. Exécuter et consulter la fenêtre du journal :

```
%put &sysdate;
%put &sysday;
%put &_automatic_;
```

La dernière commande liste toutes les macros variables du système. Ces macros variables peuvent être incluses dans un programme pour dater les sorties.

`call symput`

L'instruction `call symput` permet de créer une macro-variables en lui affectant les valeurs d'une variable d'une table SAS. Cette instruction s'utilise exclusivement dans une étape DATA.

```
CALL SYMPUT("nom-macro-var",valeur-macro-var);
```

Cela signifie que le contenu de cette macro variable n'est pas prédéfini par une chaîne de caractères mais peut évoluer en cours d'exécution. Ainsi dans l'exemple :

```
data _NULL_;
   set table-sas;
   call symput("nobs",_N_);
run;
```

Comme la variable prédéfinie `_N_` est incrémenté par le numéro de l'observation courante, à l'issue de l'excécution la macro variable contient comme "valeur" le nombre d'observations de la `table-sas`. C'est évidemment très pratique quand cette information n'est pas connue a priori sur l'ensemble des tables qui seront traitées.

2.2 Macro fonction

Un ensemble de macros fonctions s'appliquant à des macros variables sont prédéfinies. En voici des exemples

```
%let semaine = lundi - mardi - mercredi - jeudi -
```

```
 vendredi - samedi - dimanche;
%let longueur = %length(&semaine);
%let jour2 = %scan(&semaine,2,'-');
```

Dans cet exemple, la macro-variable longueur contient la longueur de la chaîne de caractères semaine soit 63. La macro-variable jour2 correspond ensuite au 2ème mot de la chaîne (soit mardi). Le troisième argument indique que le séparateur est un tiret.

Par définition, une macro-variable sert à stocker du texte. Par exemple, la valeur d'une macro-variable à laquelle on affecte `1+2` est la chaîne de caractères `1+2` et non l'entier 3. Toutefois, il est possible de forcer le compilateur macro à effectuer l'opération avec la macro fonction `%eval` qui opère des soustractions, multiplications et divisions seulement sur des entiers à partir d'expressions contenant des macro-variables. Si la valeur évaluée contient des décimales, la valeur est tronquée à la partie entière.

```
%let i = 22;
%let j = &i/7; /* j contient 22/7 */
%let k = %eval(&i/7); /* k contient 3 */
```

2.3 Macro-commandes

Syntaxe d'une macro commande

La déclaration d'une macro-commande (ou *macro* tout court) suit les mêmes principes,

```
%macro impdat;
%* ceci est un commentaire;
proc print;
run;
%mend impdat;
```

elle peut inclure des paramètres qui deviennent des macro-variables locales :

```
%macro plot(yvar,xvar);
proc plot;
  plot &yvar*&xvar;
run;
%mend plot;
```

L'exécution d'une macro est invoquée en faisant précéder son nom du caractère `%` :

```
%impdat;
%plot(taille,poids);
```

Valeurs et ordre des paramètres

Les paramètres peuvent avoir des valeurs par défaut :

```
%macro plot(yvar=taille,xvar=poids);
proc plot;
  plot &yvar*&xvar;
run;
%mend plot;
```

Il n'est plus nécessaire de leur assigner des valeurs. Attention à l'ordre des paramètres qui est significatif dans le cas de paramètres sans valeurs par défaut. Sinon, le nom du paramètre est obligatoire au moment de l'appel. Par principe, les premiers paramètres d'une macro commande sont ceux les plus utilisés qui n'ont pas de valeur par défaut.

3 Exemple de déroulement élémentaire

L'objectif est d'écrire une macro-commande adaptée à la lecture de tout fichier texte contenant une première colonne d'identificateurs des observations suivie de plusieurs colonnes de variables séparées, par défaut, par des espaces.

Le fichier `temp.dat` est dans le répertoire `data` de Wikistat.

3.1 Programme

Le programme s'écrit de la façon suivante dans un cas particulier.

```
data sasuser.temp;
infile "temp.dat" ;
```

```
input ville $jan fevr mars avri mai juin juil
            aout sept octo nove dece;
run;
```

Tester le programme.

3.2 Macros-variables

Introduire les macros variables qui seront les paramètres.

```
%let fich=temp.dat;
/* attention au chemin d'accès au fichier */
%let tab=tempville;
%let ident=ville;
%let listv=jan fevr mars avri mai juin juil
  aout sept octo nove dece;
%let dlm=" ";
/* attention à l'espace */
data &tab;
infile "&fich" dlm=&dlm;
/* attention aux doubles quotes*/
input &ident $ &listev;
run;
```

Retester le programme.

3.3 Macros-commandes

Transformer le programme en une macro commande. Le paramètre `dlm` recevant une valeur par défaut.

```
%macro lecacp(in,out,ident,listev,dlm=" ");
%* ajouter des commentaires ;
%* documentant les paramètres;
data sasuser.&out;
infile "&in" dlm=&dlm;
input &ident $ &listev;
run;
%mend;
```

Exécuter enfin l'appel de la macro dans SAS afin de créer la table SAS à partir du fichier texte `/users/magist/besse/data/TP/temp.dat`.

```
%lecacp(temp.dat, temp, ville, janv fevr mars
  avri mai juin juil aout sept oct nov dec);
proc print; run;
```

La macro pourrait être complétée en ajoutant un paramètre de librairie. Par défaut la table SAS est temporaire dans la librairie de travail `work`.

Créer un fichier de nom `macros.sas` qui contiendra toutes les macros, les unes derrières les autres, couramment utilisée. Ce fichier sera exécuté au début de chaque session.

4 Quelques macros-commandes élémentaires

4.1 Représentations de courbes

Transposition et concaténation

Pour représenter les courbes de température, une transformation préalable des données est nécessaire afin de les présenter sous une forme acceptable à la procédure SAS. Le fichier à créer contient 3 variables, le nom de la ville, la température, le numéro du mois. Les températures des 32 villes sont concaténées verticalement et distinguées par leur nom qui joue le rôle d'un "facteur" dans l'utilisation de la procédure `gplot`. Noter l'utilisation de la commande `array` qui crée un vecteur comprenant un nombre, noté `c` de composantes, égal au nombre de variables numériques dans la liste de sa définition.

```
data ttemp (keep= i nom temp);
set sasuser.temp;
array c{*} _numeric_;
do i=1 to dim(c);
   nom=ville;
   temp=c{i};
   output;
   end;
run;
```

Graphe

En s'inspirant du programme ci-dessous, écrire et soumettre à SAS une macro qui reçoit en paramètre une liste de villes et trace le graphe de leurs températures moyennes mensuelles : `%gtemp("tlse" "pari" "lill").`

```
%let listvill="ajac" "lill" "bres";
proc gplot data=ttemp;
symbol  i=spline;
where nom in (&listvill);
plot temp*i=nom;
run;quit;goptions reset=all;
```

Noter que la syntaxe de la commande `where` impose de présenter les villes entre "" dans la liste. Les lignes des courbes mériteraient d'être plus épaisses.

Créer la macro commande `%gtemp` à partir du programme ci-dessus.

4.2 Création de fichiers graphiques

La macro ci-dessous est très pratique pour générer directement des fichiers jpeg, pdf ou postscript lors de la mise au point de commandes graphiques en contrôlant leur taille.

```
%macro jpeg(file,l=10,h=10,pilote=jpeg);
quit;
filename gsasfile  "&file..jpg";
goptions device=&pilote vsize=&h cm hsize=&l cm
         gend='0a'x  gaccess=gsasfile;
%mend;
```

Tester son utilisation :

```
%jpeg(tlpali);
%gtemp("tlse" "pari" "lill");
%jpeg(ajlibr);
%gtemp("ajac" "lill" "bres");
```

Vérifier la bonne création des fichiers dans la fenêtre journal.

4.3 Nuage de points

Un nuage de points à deux dimensions est produit par le programme ci-dessous

```
proc gplot data=sasuser.temp;
symbol1 v=dot i=none;
plot janv*fevr=1;
run;quit;
```

Écrire une macro `%nuage` qui

1. reçoit en paramètre le nom d'une table, le nom de 2 variables, un type de caractère (par défaut `v=dot`),
2. affiche le nuage de points croisant ces deux variables,
3. écrit un titre avec le nom des données et des variables.

Tester cette macro pour illustrer la décroissance de la corrélation avec le temps.

4.4 Macros plus complexes

Des macros SAS plus particulièrement destinées aux méthodes de statistique multidimensionnnelle (acp, afc, afrc, afcm...) et à leurs graphiques sont disponibles dans la page d'url : `http://www.math.univ-toulouse.fr/~besse/pub/sas`

Charger certaines de ces macros : `%acp.sas, %bootstrp.sas %gacpicx.sas %vcklogit.sas` et commenter les programmes.

Voici une version simplifiée de l'analyse en composantes principales.

```
%macro acp(ldataset, lident, llistev, red=);
%global dataset ident listev;
%let dataset=&ldataset;
%let ident=&lident;
%let listev=&llistev;
%* Acp de dataset ;
%* ident : variable contenant les;
%*       identificateurs des individus;
%* llistev : liste des variables (numeriques);
%* par defaut : reduites sinon red=cov;
```

```
%* options  edition;
options linesize=80 pagesize=60 number;
title "A.c.p. des donnees de &dataset";
footnote;

proc princomp data=donnees
          outstat=eltpr out=compr
          vardef=N &red;
    var &listev;
run;
%mend acp;

/* acp r'eduite par d'efaut */
%acp(crime,staten,murder--auto);
```

5 Bibliothèque de macros commandes

5.1 Dans un fichier

Une macro-commande doit être déclarée avant toute utilisation au cours d'un même session SAS. Lorsque certaines macro-commandes sont régulièrement utilisées, il est pratique de pouvoir les conserver les unes à la suite des autres dans un fichier muni de l'extension `.sas`. En les ajoutant dans un fichier appelé `macros.sas`, par exemple, il suffira ensuite dans n'importe quel programme SAS de faire l'appel :

```
%INCLUDE "macro.sas";
```

5.2 Dans un répertoire

Il est aussi possible de stocker les macros dans une *bibliothèque* de macros spécifique à un utilisateur ou à un groupe. Les règles suivantes sont à respecter :

1. Chaque déclaration de macro et une seule est enregistrée dans un fichier qui a pour nom l'identificateur de cette macro suivi de l'extension `.sas` : `impdat.sas`,
2. Tous ces fichiers sont regroupés dans un ou des répertoires qui constituent la ou les bibliothèques,
3. le ou les chemins d'accès sont spécifiés une fois pour toute dans le fichier de configuration SAS de l'utilisateur. Encore faut-il avoir identifié et localisé ce fichier et pouvoir y avoir accès !

En début de chaque session, SAS charge le répertoire de macros.