Formation SAS Macros – Cas pratiques

Martin Chevalier (DMCSI) et Yves Dubois (DESE)

18-19 janvier 2016

Les fichiers de la formation sont stockés sur le lecteur réseau `\\s90sfer1\formation\`, dans le sous-dossier `A_Salle_401\Sas_formation_20160118`.

1 Principes fondamentaux du macro-langage

Cas pratique 1.1 Afficher les macro-variables système

1. En utilisant la macro-instruction `%PUT` avec le mot-clé `_ALL_`, affichez l'ensemble des macro-variables système dans le journal. Repérez les macro-variables contenant d'une part la version de SAS utilisée et d'autre part l'identifiant utilisateur (votre idep).
2. Utilisez ces deux macro-variables et la macro-instruction `%PUT` pour afficher dans le journal :

   ```
   Version de SAS utilisée : [version]
   Identifant utilisateur : [id_utilisateur]
   ```

3. À l'aide de la macro-instruction `%LET`, affectez la valeur `Hello world!` à la macro-variable `maMacroVar`. Affichez-la dans le journal.
4. Affectez la valeur `NOTE: Hello world!` (en respectant la casse) à la variable `maMacroVar2` et affichez-la dans le journal. Que remarquez-vous ? Que se passe-t-il à votre avis si le texte d'une macro-variable commence par `WARNING:` ou `ERROR:` ?
5. Utiliser le mot-clé `_USER_` pour afficher les macro-variables définies par vous-mêmes depuis le lancement de la session.

Proposition de solution

```
/*1.*/
%PUT _ALL_;
/*Les macro-variables stockant la version de SAS et l'identifiant
utilisateur sont respectivement &sysver et &sysuserid.*/

/*2.*/
%PUT Version de SAS utilisée : &sysver;
%PUT Identifiant utilisateur : &sysuserid;

/*3.*/
%LET maMacroVar = Hello world!;
%PUT &maMacroVar;

/*4.*/
%LET maMacroVar2 = NOTE: Hello world!;
%PUT &maMacroVar2;
/*Quand une macro-variable commence par NOTE:, WARNING: ou ERROR: elle
apparaît dans le journal avec la couleur des notes, des avertissements
et des erreurs respectivement.*/
%PUT WARNING: Hello world!;
%PUT ERROR: Hello world!;

/*5.*/
%PUT _USER_;
```

Cas pratique 1.2 Centraliser des paramètres en début de code

1. Affectez à la macro-variable `dossierFormation` le chemin du dossier contenant les fichiers de la formation. Vérifiez qu'elle est bien définie en affichant sa valeur dans le journal avec la macro-instruction `%PUT`.

2. Définissez la librairie `form` qui pointe vers le dossier contenant les fichiers de la formation en utilisant la macro-variable `dossierFormation`. Prenez garde à bien utiliser des guillemets doubles `""` et non simples `''`.
3. À l'aide d'une **`PROC EXPORT`** et toujours en utilisant la macro-variable `dossierFormation`, exportez la table `sashelp.shoes` sous le nom `shoes_[id_utilisateur].csv` dans le sous-dossier `shoes` de la formation. Pensez à utiliser la macro-variable système codant l'identifiant utilisateur.

Proposition de solution

```
/*1.*/
%LET dossierFormation =
   "\\s90sfer1\formation\A_Salle_401\Sas_formation_20160118";
%PUT &dossierFormation;

/*2.*/
LIBNAME form '&dossierFormation'; /*Ne fonctionne pas*/
LIBNAME form "&dossierFormation"; /*Fonctionne*/

/*3.*/
PROC EXPORT
    DATA = sashelp.shoes
    OUTFILE = "&dossierFormation.\shoes\shoes_&sysuserid..xls"
    REPLACE
;
RUN;
```

Cas pratique 1.3 Déterminer la longueur d'une macro-variable et son nombre de « mots »

1. Assignez la valeur `11 21 22 91` à la macro-variable `listeReg`. Vérifiez que l'affectation a fonctionné en affichant sa valeur dans le journal.
2. Déterminez la longueur de cette macro-variable en utilisant la macro-fonction `%LENGTH()`.
3. Utilisez la fonction `COMPRESS()` *via* la macro-fonction `%SYSFUNC()` pour supprimer les espaces dans la macro-variable `listeReg`. Déterminez la longueur de la chaîne ainsi obtenue.
4. En reprenant les éléments des questions précédentes, proposez une méthode pour déterminer automatiquement le nombre de « mots » (au sens de morceaux de texte séparés par un espace) que contient `listeReg`. Quelles sont les limites de cette méthode ?
5. Utilisez directement la fonction `COUNTW()` pour déterminer le nombre de « mots » que contient `listeReg`.

Proposition de solution

```
/*1.*/
%LET listeReg = 11 21 22 91;
%PUT Valeur de listeReg : &listeReg;

/*2.*/
%PUT Longueur de listeReg : %LENGTH(&listeReg) caractères;
```

```
/*3.*/
%LET listeRegSansEspace = %SYSFUNC(COMPRESS(&listeReg));
%PUT &listeRegSansEspace;
%PUT Longueur de listeReg sans espace : %LENGTH(&listeRegSansEspace);

/*4.*/
%LET nbMot = %SYSEVALF(%LENGTH(&listeReg) -
   %LENGTH(%SYSFUNC(COMPRESS(&listeReg))) + 1);
%PUT Nombre de mots dans listeReg : &nbMot;

/*5.*/
%LET nbMot2 = %SYSFUNC(COUNTW(&listeReg));
%PUT Nombre de mots dans listeReg : &nbMot2;
```

Cas pratique 1.4 Afficher le temps mis par une série d'opérations
La fonction `time()` de l'étape **DATA** renvoie l'heure dans la journée en secondes écoulées depuis minuit.

1. À l'aide de l'instruction `%PUT`, affichez le nombre de secondes écoulées depuis minuit dans le journal.
2. Stockez le nombre de secondes écoulées depuis minuit dans la macro-variable `debut` et affichez-le à plusieurs reprises. Que constatez-vous ?
3. En utilisant de nouveau la fonction `time()` et la macro-fonction `%SYSEVALF()`, affichez le nombre de secondes écoulées depuis que la macro-variable `debut` a été créée.
4. Utilisez la fonction `ROUND()` pour arrondir cette valeur à 0,01 seconde près.

Proposition de solution

```
/*1.*/
%PUT %SYSFUNC(time());

/*2.*/
%LET debut = %SYSFUNC(time());
%PUT &debut;
%PUT &debut;
/*La macro-variable &debut stocke le nombre de secondes après minuit
au moment de sa création : elle contient toujours la même valeur, peu
importe le moment où on l'affiche.*/

/*3.*/
%PUT Nombre de secondes écoulées : %SYSEVALF(%SYSFUNC(time()) - &debut);

/*4.*/
%PUT Nombre de secondes écoulées :
   %SYSFUNC(ROUND(%SYSEVALF(%SYSFUNC(time()) - &debut),0.01));
```

2 Construire des macro-programmes

Préliminaires :

– Le sous-dossier `Enquete_emploi` contient des extractions (1 000 individus) des tables trimestrielles de l'enquête Emploi en continu de 2008 à 2012. Dans la suite des cas pratique, la bibliothèque correspondante est appelée `eec`. Chaque table comporte une sélection de 6 variables :

identifiant : l'identifiant de l'individu enquêté

sexe : `"1"` homme, `"2"` femme.

matri : statut matrimonial. `"1"` célibataire, `"2"` marié(e), `"3"` veuf(ve), `"4"` divorcé(e).

age : âge de l'individu.

act : situation à l'égard de l'emploi au sens du Bureau international du travail. `"1"` emploi, `"2"` chômage, `"3"` inactif.

pond/pondp : pondération, `pond` pondération définitive, `pondp` pondération provisoire (en 2012 seulement).

– Le sous-dossier `RP` contient des extractions (1 000 logements) des tables régionales du recensement. Dans la suite des cas pratique, la bibliothèque correspondante est appelée `rp`. Chaque table comporte notamment :

c_geo : le code géographique du logement ;

stocd : le statut d'occupation détaillé du logement. La modalité `"10"` désigne les logements occupés par leur propriétaire.

Cas pratique 2.1 Calcul du taux de chômage à partir de l'enquête Emploi en continu
Pour l'instant, on ne tient pas compte des pondérations dans l'enquête Emploi en continu.

1. Sans macro, écrire un programme qui affiche le taux de chômage (nombre de chômeurs sur nombre d'actifs) du premier trimestre 2008.
2. Créer un macro-programme qui prend comme paramètre une année et un trimestre et qui affiche le taux de chômage pour cette année et ce trimestre. Tester le macro-programme.
3. Améliorer le programme précédent pour qu'il fonctionne même si on déclare l'année avec deux chiffre (08 pour 2008). *On utilisera une condition* `%IF` *dans la macro.* Tester le macro-programme.
4. Améliorer le programme précédent : il prendra un paramètre supplémentaire et facultatif pouvant contenir en option une liste de variable pour calculer le chômage par catégories (par âge, par sexe, par statut matrimonial, par âge*sexe ...). Par défaut, le programme calculera le chômage par âge.
5. Essayer une par une les différentes options de débogage sur la macro précédente. *Options* `MPRINT`, `MLOGIC`, `SYMBOLGEN`.

Indication : Voici une solution (parmi d'autres) à la question 1

```
DATA chomage_indiv;
    SET eec.eec20081;
    IF act="1" OR act="2" THEN ind_actif=1;
    IF act="2" THEN ind_au_chomage=1;
RUN;

PROC SUMMARY DATA=chomage_indiv nway;
    VAR ind_actif ind_au_chomage;
    OUTPUT OUT=chomage_agrege SUM=ind_actif ind_au_chomage;
RUN;

TITLE "Chomage au T1 2008";
DATA _NULL_;
    FILE PRINT;
    SET chomage_agrege;
    taux_chomage = ROUND(ind_au_chomage / ind_actif * 100,0.01);
    PUT "Taux de chomage : " taux_chomage "%";
RUN;
TITLE;
```

Proposition de solution

```
%MACRO chomage(annee,trimestre,strate=age);
    %IF(%LENGTH(&annee) = 2) %THEN %DO;
        %LET annee= %EVAL(2000 + &annee);
    %END;
    DATA chomage_indiv;
        SET eec.eec&annee.&trimestre.;
        IF act = "1" or act = "2" THEN ind_actif = 1;
        IF act = "2" THEN ind_au_chomage=1;
    RUN;

    PROC SUMMARY DATA=chomage_indiv nway;
        var ind_actif ind_au_chomage;
        class &strate;
        OUTPUT OUT = chomage_agrege SUM = ind_actif ind_au_chomage;
    RUN;

    TITLE "Chomage au T&trimestre &annee";
    DATA _NULL_;
        FILE PRINT;
        SET chomage_agrege;
        taux_chomage = ROUND(ind_au_chomage / ind_actif * 100,0.01);
        PUT "Taux de chomage &strate " &strate " : " taux_chomage "%";
    RUN;
    TITLE;
%MEND;

%chomage(2008,1);
%chomage(08,1,strate=matri sexe);
%chomage(10,4,strate=);

OPTIONS MPRINT SYMBOLGEN MLOGIC;
%chomage(10,4,strate=);
OPTIONS NOMPRINT NOSYMBOLGEN NOMLOGIC;
```

Cas pratique 2.2 Calcul d'une série de chômage

1. Écrire un macro-programme qui affiche une série de taux de chômage par trimestre. La macro aura deux paramètres, une année de début et une année de fin. Les années pourront être données avec deux chiffres ou quatre. *On utilisera deux boucles* `%DO` *imbriquées* .
2. Améliorer le programme précédent mais cette fois-ci en tenant compte des pondérations. Attention, le nom de la variable de pondération est différent selon l'année : En 2012, la variable de pondération porte un nom différent car les poids calculés sont encore provisoires.
3. On ajoutera un paramètre à la macro pour indiquer la dernière année définitive.
4. Stocker le code généré par le macro-programme précédent dans une macro variable (on utilisera `%NRBQUOTE`) et l'afficher dans la LOG avec un `%PUT`). Que s'est-il passé ? *Le macro-processeur génère du texte, suivant le contexte dans lequel est appelée la macro, celui-ci sera exécutée comme du code, stockée dans une variable, une macro-variable....*

Proposition de solution

```
%MACRO chomage(anneedeb,anneefin,anneedef);
    %IF(%LENGTH(&anneedeb) <= 2) %THEN %DO;
        %LET anneedeb= %SYSEVALF(2000 + &anneedeb);
    %END;
    %IF(%LENGTH(&anneefin) <= 2) %THEN %DO;
        %LET anneefin= %SYSEVALF(2000 + &anneefin);
    %END;
    %DO annee = &anneedeb %TO &anneefin;
        %DO trimestre = 1 %TO 4;
            DATA chomage_indiv&annee.&trimestre.;
                SET eec.eec&annee.&trimestre.;
                IF act = "1" or act="2" THEN ind_actif = 1;
                IF act = "2" THEN ind_au_chomage = 1;
                date="&annee.T&trimestre.";
                %IF(&annee > &anneedef) %THEN %DO;
                    poids = pondp;
                %END;
                %ELSE %DO;
                    poids = pond;
                %END;
            RUN;
        %END;
    %END;

    DATA chomage_indiv;
        SET
        %DO annee = &anneedeb %TO &anneefin;
            %DO trimestre = 1 %TO 4;
            chomage_indiv&annee.&trimestre.
            %END;
        %END;
        ;   /* Attention au point virgule !!! */
    RUN;
    PROC SUMMARY DATA=chomage_indiv nway;
        var ind_actif ind_au_chomage;
        weight poids;
        class date;
        OUTPUT OUT = chomage_agrege SUM = ind_actif ind_au_chomage;
```

```
    RUN;

    TITLE "Chomage de &anneedeb à &anneefin";
    DATA _NULL_;
        FILE PRINT;
        SET chomage_agrege;
        taux_chomage = ROUND(ind_au_chomage / ind_actif * 100,0.01);
        PUT "Taux de chomage " date " : " taux_chomage "%";
    RUN;
    TITLE;
%MEND;

%chomage(2008,2012,2011);
%chomage(08,12);
%chomage(08,12);

%PUT %NRBQUOTE(%chomage(2008,2012,2011));
```

Cas pratique 2.3 Panel : Chômage des jeunes

1. Écrire un macro-programme qui crée une table de panel sur les individus ayant entre 25 et 29 ans : on empilera les tables trimestrielles avec un SET. La macro prendra deux paramètres : l'année de début et l'année de fin du panel. La table créée aura outre les variables initiales, une variable contenant la date à laquelle le programme est exécuté, une autre l'année de l'enquête, et une troisième le trimestre.
2. On veut améliorer le programme précédent en donnant la possibilité pour l'utilisateur de choisir un sous-champ. Par exemple il pourra appeler la macro avec les paramètres suivants : `%choj(2008,2011,champ=(sexe=1 and matri=1));`. Il faudra tenir compte du cas particulier où il n'y a pas de condition indiquée.
3. Modifier la macro précédente afin qu'elle affiche le temps d'exécution total de la macro.

Proposition de solution

```
%MACRO chomage_jeunes(anneedeb,anneefin,anneedef,champ=);
    %LET debtime_macro = %SYSFUNC(time());
    %IF(%LENGTH(&anneedeb) <= 2) %THEN %DO;
        %LET anneedeb= %SYSEVALF(2000 + &anneedeb);
    %END;
    %IF(%LENGTH(&anneefin) <= 2) %THEN %DO;
        %LET anneefin= %SYSEVALF(2000 + &anneefin);
    %END;
    %DO annee = &anneedeb %TO &anneefin;
        %DO trimestre = 1 %TO 4;
            DATA chomage_indiv&annee.&trimestre.;
                SET eec.eec&annee.&trimestre.;
                WHERE "25" <= age < "30";
                %IF %LENGTH(&champ) > 0 %THEN %DO;
                    IF &champ;
                %END;
                annee="&annee";
                trimestre="&trimestre";
                %IF(&annee > &anneedef) %THEN %DO;
                    poids = pondp;
                    DROP pond;
```

```
                %END;
                %ELSE %DO;
                    poids = pond;
                    DROP pond;
                %END;
            RUN;
        %END;
    %END;

    DATA chomage_panel;
        SET
        %DO annee = &anneedeb %TO &anneefin;
            %DO trimestre = 1 %TO 4;
                chomage_indiv&annee.&trimestre.
            %END;
        %END;
        ;
        BY identifiant;
    RUN;
    %LET duree_exec_macro = %SYSEVALF(%sysfunc(time()) - &debtime_macro);
    %PUT Durée d execution de a macro :
       %SYSFUNC(ROUND(&duree_exec_macro,0.1)) secondes;
%MEND;

%chomage_jeunes(08,09,2011);
%chomage_jeunes(08,09,2011,champ=age<"27");
```

Cas pratique 2.4 Utiliser un catalogue de macros externe
Le sous-dossier `calmar` contient le catalogue de macro compilées de calage sur marges `%calmar` disponible sur le site internet de l'Insee.

1. Créez la bibliothèque qui pointe vers le sous-dossier `calmar`. En utilisant l'option `SASMSTORE`, assignez à cette bibliothèque le chemin de recherche par défaut des macros.
2. Dans l'explorateur de fichiers de SAS, déplacez-vous jusque dans la bibliothèque que vous avez créée et parcourez le catalogue de macros compilées qui doit normalement s'y trouver.
3. Lancer la macro `%calmar` (sans argument). Que se passe-t-il ? Avez-vous réussi à lancer le programme dans le catalogue externe de macro compilées ?
4. Dans votre éditeur, créez le macro-programme `%calmar` sans paramètre et dont la seule fonction est d'afficher `Macro-programme dans la WORK` dans le journal. Lancez de nouveau le macro-programme `%calmar`. Que se passe-t-il ? Qu'est-ce que cela vous apprend sur le fonctionnement de l'option `SASMSTORE` ?

Proposition de solution

```
/*1.*/
LIBNAME calmar
   "\\s90sfer1\formation\A_Salle_401\Sas_formation_20160118\calmar";
OPTIONS SASMSTORE = calmar;

/*3.*/
%calmar();
/*Le macro-programme rencontre une erreur mais s'exécute: on a bien réussi
```

```
à utiliser le catalogue externe de macros.*/

/*4.*/
%MACRO calmar();
    %PUT Macro-programme dans la WORK;
%MEND calmar;
%calmar();
/*C'est le macro-programme compilé dans la WORK qui se lance : malgré
l'option SASMSTORE, les macro-programmes compilés dans la WORK restent
"prioritaires".*/
```

3 Travailler avec des tableaux et le CALL SYMPUT

Cas pratique 3.1 Scinder une table selon les modalités d'une variable
Objectif : Créer une macro pour scinder la table `rp.r11` selon le code géographique (variable `c_geo`). La macro ne prendra aucun paramètre.

1. Dans un premier temps la macro utilisera :

   ```
   PROC SORT DATA = rp.r11(KEEP=c_geo) NODUPKEY OUT = liste_c_geo;
       BY c_geo;
   RUN;
   ```

   pour créer une table contenant seulement les codes géographiques avec une ligne par code.
2. Puis dans une étape **DATA** elle générera un tableau de macro-variables.
3. Enfin elle bouclera sur ce tableau pour créer une table par code géographique.

Proposition de solution

```
%MACRO scinder_table();
    /* On crée la table liste_c_geo contenant une ligne par modalité
    de la variable c_geo */
    PROC SORT DATA=rp.r11(KEEP=c_geo) NODUPKEY OUT=liste_c_geo;
        BY c_geo;
    RUN;

    /* On crée un tableau de macro-variables codes_geo
    La taille du tableau est conservée dans la macro-variable codes_geo_nb
       */
    DATA liste_c_geo;
        CALL SYMPUTX(CAT("codes_geo",_N_), memname, "L");
        CALL SYMPUTX("codes_geo_nb",_N_, "L");
    RUN;

    /* On crée les tables par code géo en ventilant par zone
       géographique.*/
    %DO i = 1 %TO &codes_geo_nb;
        DATA liste_c_geo;
            SET rp.r11;
            WHERE c_geo="&&codes_geo&i";
        RUN;
    %END;

%MEND;

%scinder_table();
```

Cas pratique 3.2 Empiler toutes les tables d'une librairie (utilisation : `sashelp.vtable`)

1. Créez une macro qui empile toutes les tables de l'enquête Emploi en continu contenues dans la bibliothèque `eec` dans une table `panel_eec`. On utilisera la table `sashelp.vtable` pour créer la liste des tables.
2. Comme empiler des tables d'une bibliothèque est une chose que l'on utilise très souvent, on rend cette macro plus générique. On la modifie pour qu'elle prenne en paramètres :

– le nom de la bibliothèque dont on veut empiler les tables ;
– le nom de la table dans laquelle elles seront empilées ;
– un paramètre optionnel `champ` pour indiquer éventuellement un champ.

Exemple d'utilisation de cette nouvelle macro :

```
%empile_lib(rp, RP_FRANCE);
%empile_lib(eec, PANEL_EEC);
%empile_lib(eec, PANEL_EEC_JEUNES, champ= (25<=age<=30));
```

Proposition de solution

```
%MACRO empile_lib(lib,OUT,champ=);
    /* On crée un tableau de macro-variables mestables */
    DATA _NULL_;
        SET sashelp.vtable;
        WHERE libname=upcase("&lib.");
        CALL SYMPUTX(CAT("mestables",_N_), memname);
        CALL SYMPUTX("mestables_nb",_N_);
    RUN;

    /* On crée la table de sortie ... */
    DATA &OUT.;
        /* En empilant toutes les tables de la librairie */
        SET
            %DO i = 1 %TO &mestables_nb;
                &lib..&&mestables&i
            %END;
        ;
        /* S'il y a un filtre de champ indiqué on applique ce filtre */
        %IF(%LENGTH(&champ) > 0) %THEN %DO;
            WHERE &champ;
        %END;
    RUN;
%MEND;
```

Cas pratique 3.3 Renommer toutes les variables d'une table (utilisation : `sashelp.vcolumn`)

Objectif : Copier la table `eec.eec20081` dans une nouvelle table en renommant les variables :
– les variables numériques seront préfixées avec num_
– les variables caractères seront préfixées avec car_

Proposition de solution

```
%MACRO rename_columns();
    DATA _NULL_;
        SET sashelp.vcolumn;
        WHERE libname='EEC' and memname='EEC20081';
        CALL SYMPUTX(CAT("nom_colonne",_N_), name,"L");
        CALL SYMPUTX(CAT("type_colonne",_N_), type,"L");
        CALL SYMPUTX("nombre_colonnes",_N_);
    RUN;

    DATA eec20081_rename;
        SET eec.eec20081;
        %DO i = 1 %TO &nombre_colonnes;
```

```
            %IF(&&type_colonne&i = num) %THEN %DO;
                RENAME  &&nom_colonne&i = num_&&nom_colonne&i;
            %END;
            %ELSE %DO;
                RENAME  &&nom_colonne&i = car_&&nom_colonne&i;
            %END;
        %END;
        ;
    RUN;
%MEND;

%rename_columns();
```

4 Travailler avec des listes et la PROC SQL

Cas pratique 4.1 Compter les observations d'une table
La table `sashelp.shoes` contient des informations sur une entreprise fictive commercialisant des chaussures.

1. À l'aide d'une instruction SQL utilisant la fonction `COUNT()`, comptez le nombre d'observations de cette table et assignez-le à la macro-variable `nbObs`.
2. À l'aide de de la structure `COUNT(DISTINCT variable)`, créez la macro-variable `nbRegion` contenant le nombre de régions dans lesquelles l'entreprise est implantée.
3. Assignez à la macro-variable `nbObsAfrica` le nombre d'observations relatives à l'Afrique. Utilisez deux méthodes différentes (avec un `WHERE` ou la fonction `SUM()`).
4. En utilisant les mots-clés `SEPARATED BY` et `GROUP BY`, assignez à la macro-variable `listeNbObs` la liste (séparée par des espaces) du nombre d'observations associées à chaque région.
5. (Avancé) En utilisant des fonctions de manipulation de chaînes de caractères dans l'instruction SQL, faites en sorte que la valeur de `listeNbObs` soit du type :

   `Africa = 56, Asia = 14, ...`

Proposition de solution

```
/*1.*/
PROC SQL NOPRINT;
    SELECT COUNT(*) INTO :nbObs FROM sashelp.shoes;
QUIT;
%PUT &nbObs;

/*2.*/
PROC SQL NOPRINT;
    SELECT COUNT(DISTINCT region) INTO :nbRegion FROM sashelp.shoes;
QUIT;
%PUT &nbRegion;

/*3.*/
PROC SQL NOPRINT;
    SELECT COUNT(*) INTO :nbObsAfrica FROM sashelp.shoes WHERE region =
       'Africa';
QUIT;
%PUT &nbObsAfrica;
PROC SQL NOPRINT;
    SELECT SUM(region = 'Africa') INTO :nbObsAfrica FROM sashelp.shoes;
QUIT;
%PUT &nbObsAfrica;

/*4.*/
PROC SQL NOPRINT;
    SELECT COUNT(*) INTO :listeNbObs SEPARATED BY ' '
    FROM sashelp.shoes GROUP BY region
    ;
QUIT;
%PUT &listeNbObs;
```

```
/*5.*/
PROC SQL NOPRINT;
    SELECT CAT(region,' = ',COUNT(*)) INTO :listeNbObs SEPARATED BY ', '
    FROM sashelp.shoes GROUP BY region
    ;
QUIT;
%PUT &listeNbObs;

/*On supprime les espaces en trop avec TRIM() et COMPRESS() :*/
PROC SQL NOPRINT;
    SELECT CAT(TRIM(region),' = ',COUNT(*)) INTO :listeNbObs SEPARATED BY
        ', '
    FROM sashelp.shoes GROUP BY region
    ;
QUIT;
%PUT &listeNbObs;
```

Cas pratique 4.2 Renommer en bloc les variables d'une table
L'objectif de ce cas pratique est de créer le programme `%renommerVariable(tableEntree =, tableSortie = )` qui renomme automatiquement toutes les variables de la table `&entree` avec les noms `var1`, `var2`, ... pour faciliter des manipulations ultérieures. La table `sashelp.shoes` servira à tester le programme.

1. Dans un premier temps, créez manuellement (avec un `%LET`) les macro-variables `listeVariable` et `nombreVariable` qui contiennent respectivement la liste et le nombre de variables de la table `sashelp.shoes`. À l'aide de ces macro-variables, d'une boucle `%DO %TO` et de la macro-fonction `%SCAN()`, écrivez une première version du macro-programme `%renommerVariable()`. Utilisez l'option `MPRINT` pour afficher le code effectivement soumis.
2. En utilisant la table `dictionary.columns`, écrivez la requête SQL qui affiche la liste des variables de la table `sashelp.shoes` dans la fenêtre de résultats. Utilisez cette requête et les mots-clés `INTO` et `SEPARATED BY` pour créer automatiquement les macro-variables `listeVariable` et `nombreVariable`.
3. À partir des deux questions précédentes, construisez une version entièrement automatique de `%renommerVariable()`. Vous pouvez utiliser la macro-fonction `%SCAN()` pour décomposer le chemin vers la table `biblio.nom` en `biblio` et `nom`.

Proposition de solution

```
/*1.*/
%MACRO renommerVariable(tableEntree = sashelp.shoes, tableSortie = shoes);

    /*Définition manuelle des macro-variables listeVariable et
       nombreVariable*/
    %LET listeVariable = Inventory Product Region Returns Sales Stores
       Subsidiary;
    %LET nombreVariable = 7;

    /*Etape DATA pour renommer*/
    DATA &tableSortie;
        SET &tableEntree;
        RENAME
            %DO numeroVariable = 1 %TO &nombreVariable;
```

```
                %LET nomVariable = %SCAN(&listeVariable,&numeroVariable);
                &nomVariable = var&numeroVariable
            %END;
        ;
    RUN;

%MEND;
OPTIONS MPRINT;
%renommerVariable;

/*2.*/
PROC SQL;
    SELECT name
    FROM dictionary.columns
    WHERE libname = 'SASHELP' AND memname = 'SHOES'
    ;
QUIT;

/*Création automatique des macro-variables*/
PROC SQL NOPRINT;
    SELECT name, COUNT(*)
    INTO :listeVariable SEPARATED BY ' ', :nombreVariable
    FROM dictionary.columns
    WHERE libname = 'SASHELP' AND memname = 'SHOES'
    ;
QUIT;
%PUT &listeVariable;
%PUT &nombreVariable;

/*3.*/
%MACRO renommerVariable(tableEntree = sashelp.shoes, tableSortie = shoes);

    /*Décomposition de &entree en libname et en memname pour lire
       dictionary.columns*/
    %LET libname = %UPCASE(%SCAN(&tableEntree,1,.));
    %LET memname = %UPCASE(%SCAN(&tableEntree,2,.));

    /*Définition automatique des macro-variables listeVariable et
    nombreVariable à partir de dictionary.columns*/
    PROC SQL NOPRINT;
        SELECT name, COUNT(*)
        INTO :listeVariable SEPARATED BY ' ', :nombreVariable
        FROM dictionary.columns
        WHERE libname = "&libname" AND memname = "&memname"
        ;
    QUIT;

    /*Etape DATA pour renommer*/
    DATA &tableSortie;
        SET &tableEntree;
        RENAME
            %DO numeroVariable = 1 %TO &nombreVariable;
                %LET variable = %SCAN(&listeVariable,&numeroVariable);
                &variable = var&numeroVariable
            %END;
        ;
    RUN;
```

```
%MEND;
OPTIONS MPRINT;
%renommerVariable;

/*Utilisation sur une autre table*/
%renommerVariable(tableEntree = sashelp.company,tableSortie = company);
```

Cas pratique 4.3 Travailler avec une liste de tables
Les fichiers contenus dans le sous-dossier `rp` sont des extractions des fichiers logements régionaux du recensement de la population 2012. L'objectif de ce cas pratique est de les rassembler automatiquement dans une seule table.

1. Définissez la bibliothèque `rp` correspondant au sous-dossier `rp`. En utilisant la table `dictionary.members`, écrivez la requête SQL qui affiche la liste des tables de la bibliothèque `rp`. Utilisez cette requête pour créer automatiquement les macro-variables `listeTable` et `nombreTable` contenant respectivement la liste des noms et le nombre de tables dans la bibliothèque `rp`.
2. Utilisez une boucle (dans un macro-programme) pour rassembler dans une même table tous les logements occupés par des propriétaires (`stocd = '10'`). Pensez à utiliser l'option `MPRINT` pour afficher le code effectivement soumis.
3. (Avancé) En partant de la table `dictionary.columns` et en utilisant des fonctions de manipulation de chaînes de caractères, générez directement (sans boucle) l'instruction `SET` utilisée à la question précédente. Inspirez-vous de la dernière question du cas pratique 4.1.

Proposition de solution

```
/*1.*/
PROC SQL;
    SELECT memname
    FROM dictionary.members
    WHERE libname = 'RP'
    ;
QUIT;

/*Création de listeTable et nombreTable*/
PROC SQL NOPRINT;
    SELECT memname, COUNT(memname)
    INTO :listeTable SEPARATED BY ' ', :nombreTable
    FROM dictionary.members
    WHERE libname = 'RP'
    ;
QUIT;
%PUT &listeTable;
%PUT &nombreTable;

/*2.*/
%MACRO boucle;
    DATA resultats;
        SET
            %DO numeroTable = 1 %TO &nombreTable;
                %LET nomTable = %SCAN(&listeTable,&numeroTable);
                rp.&nomTable.
            %END;
        ;
```

```
    RUN;
%MEND;
OPTIONS MPRINT;
%boucle;

/*3.*/
PROC SQL NOPRINT;
    SELECT CAT('rp.',memname)
    INTO :code SEPARATED BY ' '
    FROM dictionary.members
    WHERE libname = 'RP'
    ;
QUIT;
%PUT &code;
DATA resultats;
    SET &code;
RUN;
```

Cas pratique 4.4 Construire un programme de dichotomisation automatique
Il est parfois nécessaire dans certains contextes de dichotomiser une variable qualitative en créant une série de variable indicatrices (une par modalité de la variable qualitative) qui prennent la valeur 1 si l'observation présente la modalité en question et 0 sinon.
On se propose ici d'écrire le macro-programme `%dicho(tableEntree =, tableSortie =, variable =)` qui automatise cette opération. Afin de simplifier le problème, on se limite à la dichotomisation de variables de type caractère. La variable `region` de la table `sashelp.shoes` servira à tester le programme. Les variables indicatrices auront des noms de la forme : `region1`, `region2`, `...`

1. Écrivez « à la main » les premières instructions de dichotomisation dans une étape **DATA**.
2. À l'aide d'une instruction SQL et du mot-clé `DISTINCT`, affichez dans la fenêtre de résultats les modalités distinctes de la variable `region`.
3. Adaptez l'instruction de la question précédente pour créer les macro-variables `listeModalite` et `nombreModalite` qui stockent respectivement les valeurs et le nombre de modalités de la variable `region`. Choisissez un délimiteur adéquat pour être en mesure de parcourir `listeModalite` avec la macro-fonction `%SCAN()`.
4. Construisez le macro-programme `%dicho()` qui s'appuie sur les macro-variables `listeModalite` et `nombreModalite` et une boucle `%DO`.

Proposition de solution

```
/*1.*/
DATA shoes;
    SET sashelp.shoes;
    region1 = (region = 'Africa');
    region4 = (region = 'Central America/Caribbean');
RUN;
/*A noter : des espaces et des caractères spéciaux apparaissent
dans la valeur des modalités de la variable region.*/

/*2.*/
PROC SQL;
    SELECT DISTINCT region FROM sashelp.shoes;
QUIT;
```

```
/*3.*/
PROC SQL NOPRINT;
    SELECT DISTINCT region, COUNT(DISTINCT region)
    INTO :listeModalite SEPARATED BY '$$$', :nombreModalite
    FROM sashelp.shoes
    ;
QUIT;
%PUT &listeModalite;
%PUT &nombreModalite;
/*Le délimiteur est une chaîne de caractère complexe qui est absente
des modalités des variables à dichotomiser. Cela permet de parcourir
la chaîne de caractère avec un %SCAN() en conservant les espaces.*/
%PUT %SCAN(&listeModalite,4,$$$);

/*4.*/
%MACRO dicho(tableEntree = sashelp.shoes, tableSortie = shoes, variable =
   region);

    /*Construction automatique de &listeModalite et &nombreModalite*/
    PROC SQL NOPRINT;
        SELECT DISTINCT &variable, COUNT(DISTINCT &variable)
        INTO :listeModalite SEPARATED BY '$$$', :nombreModalite
        FROM &tableEntree
        ;
    QUIT;

    /*Dichotomisation*/
    DATA &tableSortie;
        SET &tableEntree;
        %DO numeroModalite = 1 %TO &nombreModalite;
            %LET valeurModalite =
               %SCAN(&listeModalite,&numeroModalite,$$$);
                &variable.&numeroModalite = (&variable =
                   "&valeurModalite");
                LABEL &variable.&numeroModalite = "&valeurModalite";
                /*On donne à l'indicatrice la valeur de la modalité comme
                   label*/
        %END;
    RUN;

%MEND;
OPTIONS MPRINT;
%dicho;
%dicho(variable=subsidiary);
OPTIONS NOMPRINT;
```