BIBLIOTHEQUE DE DOCUMENTATION INTERNATIONALE CONTEMPORAINE

UNIVERSITES DE PARIS

6, allée de l'Université 92001
Nanterre

Introduction à SAS

Langage, procédures, macro - langage

A. G. Hathout
Email : hathout@u-paris10.fr

TABLE DES MATIERES

➢ Avant propos

SAS est un système d'informatique décisionnelle dont les composantes procèdent à différentes tâches associées à la prise de décision (gestion de données statistiques et textuelles, analyses descriptives, modélisation, discrimination, ...).

Puisque le terme de « décision » et l'acte mental qui lui est propre débordent des considérations techniques approchées dans ces pages, j'ai proposé sur le site documentaire intranet.u-paris10.fr de l'université de Paris X – Nanterre la narration d'une une suite d'événements survenus entre 1900 et 2000 dont l'examen devrait aider le praticien de l'analyse des données à inscrire la genèse de l'idée d'informatique décisionnelle dans le large contexte de l'information et de la société.

L'essentiel de ce texte a été rédigé entre 1993 et 1999 pour servir de support de cours à des étudiants de l'université de Paris X – Nanterre. Il a été revu entre mars et mai 2002 de manière à ce qu'il soit accessible à un plus large public sans intervention d'un moniteur. C'est pourquoi, certains des exercices qui figuraient dans les versions antérieures à 2000 ont disparu alors que l'explication du fonctionnement des programmes et de leur domaine d'application a été développée.

Ce support de cours a été organisé en exercices qui vont du simple au moins simple. Les programmes SAS y sont encadrés. Afin de rendre leur lecture plus aisée et leur application plus parlante, ils vont de pair avec des fichiers statistiques et textuels, disponibles en copie libre sur les ordinateurs de l'université de Paris X – Nanterre.

Le lecteur qui aimerait se familiariser à la pratique de SAS en se servant de ce support de cours tirera un meilleur profit en copiant les fichiers qui l'accompagnent, conformément aux indications de l'annexe numéro 1 (création d'un tableau de bord personnel).

Le lecteur qui connaît déjà SAS mais qui désire se familiariser avec le macro–langage peut se reporter de suite au chapitre 3 après avoir copié la bibliothèque de macro-expressions mises à sa disposition sur les ordinateurs de l'université de Paris X – Nanterre. Pour effectuer cette copie, se référer à l'annexe 1 (Création d'un tableau de bord personnel).

Chapitre I : INTRODUCTION A L'ENVIRONNEMENT SAS

Section 1 : Organisation d'un programme SAS

La partie principale d'un programme SAS est composée d'étapes. Une étape commence de l'une des deux manières suivantes :

- avec le mot clé « DATA ». Il s'agit alors de la création d'un tableau de données que nous désignerons par DATA.
- avec le mot clé « PROC quand il s'agit d'appliquer une procédure à une DATA préalablement créée par SAS dans une étape DATA.

Syntaxe des étapes DATA et PROC

Ceux qui débutent en SAS ont parfois tendance à utiliser dans une étape PROC des éléments de syntaxe qui relèvent de l'étape DATA. C'est pourquoi, il est bon de savoir dès le départ que chaque procédure de SAS a sa propre syntaxe, et qu'il faut tout oublier de la syntaxe de l'étape DATA pendant que l'on rédige une étape PROC.

La syntaxe dans une étape DATA	**La syntaxe dans une procédure**
La syntaxe de l'étape DATA est celle du langage SAS tel qu'il est décrit dans le manuel « *SAS Language* ». Voici un exemple d'étape DATA : DATA madata ; x=1 ; y=2 ; z=3 ; total = x+y+z ; RUN ; Ce programme crée une DATA SAS qui s'appelle « madata ». Cette DATA contient quatre variables : x, y, z et leur total.	La syntaxe qui prévaut dans une procédure SAS est propre à la procédure. Pour connaître la syntaxe d'une procédure précise, vous pouvez consulter le manuel « *SAS Procedures Guide* ». Voici un exemple d'étape qui concerne une procédure : PROC PRINT DATA=madata ; VAR x y z ; RUN ; Ci-avant, on invoque la procédure PRINT. Celle-ci consiste à imprimer une DATA SAS préalablement créée. La syntaxe de la procédure PRINT autorise des précisions telles que : « VAR ... liste » (variables objet de l'action d'imprimer). On observera que les noms des variables à imprimer (x, y et z) sont séparés par un ou plusieurs espaces blancs parce que PROC PRINT accepte qu'il en soit ainsi (c'est toutefois général à toutes procédures).

Section 2 : Mon premier programme SAS

La manière d'entrer dans l'environnement SAS dépend de votre installation. Une fois que vous êtes dans SAS, commencez par organiser vos fenêtres de manière rationnelle (cela dépend notamment de la dimension de votre écran et des habitudes de travail que vous allez emprunter progressivement).

Vous disposez notamment des trois fenêtres suivantes :

- « prog » où vous écrivez le programme SAS
- « log » où vous avez un compte-rendu du déroulement logique du programme
- « out » où vous avez les impressions commandées dans le programme.

A titre d'exercice,

- écrivez le programme suivant dans la fenêtre « prog »

```
/* Première étape */
DATA madata ;
x=1 ; y=2 ; z=(x+y)**2 ;
/* Deuxième étape */
PROC PRINT ;
TITLE ' Premier essai ' ;
RUN ;
```

- cliquez sur « local »
- cliquez sur « soumettre »

Le programme qui précède est composé de deux étapes : l'étape DATA définit la valeur des variables x, y et z ; alors que la seconde étape commande l'impression de la DATA qui vient d'être créée au moyen de la procédure PRINT.

La procédure PRINT aurait pu s'écrire

```
PROC PRINT DATA=madata ;
```

ce qui aurait signifié « Imprimer la DATA madata ». Mais, en l'absence de précisions quant à la DATA à imprimer, c'est la dernière DATA créée que commande le PROC PRINT.

Quand au texte compris entre le /* et le */ , c'est un commentaire.

Section 3 : Les clés de fonctions

Les clés de fonction vous permettent d'affecter une fonction particulière (soumission d'un programme à exécution par exemple) au simple fait d'appuyer sur une touche. Les affectations y sont conventionnelles

- Cliquez sur « aide»,
- puis sur « touches de fonctions »
- Procédez aux affectations suivantes :

Ces affectations sont des conventions. Vous pouvez en choisir d'autres.

Clé	affectation	signification
F3	sub	soumission d'un programme (ne fonctionne que dans la fenêtre « prog »)
F4	Pgm ; recall	aller à la fenêtre « programme » et éditer le dernier texte. Le « ; » signifie que nous concaténons les deux opérations.
F9	Clear	balayage de l'écran (fonctionne dans les trois fenêtres)

A la fin des affectations,

- Cliquez sur « enregistrer »,
- Puis sur « fin ».

Section 4 : Le mode « COMMAND LINE »

Au lieu de travailler avec la souris, vous pouvez faire apparaître dans la fenêtre PROG une ligne de commande sur laquelle vous donnez des ordres. Pour ce faire, procéder comme suit :

- Cliquez sur « général »,
- Puis sur « options »,
- Puis sur « command line».
- Au niveau « Command », écrivez « num » : les lignes de l'écran éditeur seront alors numérotées.
- Pour revenir au mode fenêtres, écrivez à nouveau « Command ».

A noter : dans les versions récentes de SAS, vous trouverez des équivalents français plus parlants. Par exemple « ligne de commande » au lieu de « Command Line ».

Ordres au niveau de la ligne de commande

Ordre	Signification
keys	Ouvrir la fenêtre des clés de fonction.
Include toto	Charger dans la fenêtre « prog » le fichier unix ‘toto.sas’
include ‘toto’	Charger dans la fenêtre « prog » le fichier unix ‘toto ‘
clear	nettoyer l’écran
rec	ou « recall » pour réediter le dernier fichier
Log	Aller à la fenêtre qui édite le déroulement logique du programme
Out	Aller à la fenêtre qui donne les résultats (output)
prog	Aller à la fenêtre programme !
change toto titi	Changer toutes les chaînes de caractères « toto » en « titi »
find carac	pour parcourir toutes les chaînes de caractère « carac »
find carac all	pour compter le nombre de fois qu’il y a la chaîne « carac » et les parcourir.
Top	Editer à partir de la ligne 1
bot	(bottom) pour éditer à partir de la dernière ligne
up	Aller plus haut
down	Aller plus bas
1	Editer à partir de la ligne n°1
36	Editer à partir de la ligne n°36
right	Aller à droite de l’écran
left	Aller à gauche de l’écran
file ‘toto’	Sauvergarder le fichier édité sous le nom ‘toto’
file toto	Ssauvegarder le fichier étudié sous le nom ‘toto.sas’

Section 5 : Opérations sur lignes numérotées

Une fois que vous avez choisi le mode « ligne de commande », vous avez aussi la possibilité de faire des opérations sur les lignes sans utiliser la souris. Pour ce faire, il faudra numéroter les lignes.

Pour numéroter les lignes, écrire « num » au niveau de la ligne de commande.

Opérations d'édition en mode « ligne de commande »

Commande (1)	Signification
i	comme « insérer » : insérer une ligne
i2	insérer 2 lignes.
D	comme « delete » ; écraser la ligne
d3	écraser 3 lignes à partir de la ligne en cours
dd	écraser un bloc de ligne. Attend un second dd
c	copier une ligne attend un « o » comme « over » ou un « a » comme « after » ou un « b » comme « before »
cc	copier un bloc de lignes . attend un deuxième cc pour annoncer la fin du bloc. attend également un « a » comme « after », ou un « b » comme « before »
r	répéter la ligne
r2	répéter la ligne 2 fois
...	...
r50	répéter la ligne 50 fois
rr	répéter un bloc de lignes attend un deuxième « rr » qui annonce la fin du bloc
m	comme « move » ; déplacer une ligne attend un « a » comme « after » ou un « o » comme « over » ou un « b » comme « before »
mm	déplacer un bloc de lignes. attend un deuxième « mm » qui annonce la fin du bloc.
cols	identifier avec une règle le numéro des colonnes

(1) A écrire dans la zone numérotée, à gauche de l'écran.

Chapitre II : EXERCICES GENERAUX

Section 1 : Introduction à l'étape DATA

Dans cet exercice est définie une étape DATA appelée par l'utilisateur « madata ». Ce type de lecture (INPUT) , de la forme :

[@colonne nom format]

équivaut à dire : je débute ma lecture en colonne « colonne » d'une variable que j'appelle « nom » et que je lis avec le format « format ». Ainsi [@1 nom $CHAR12.] signifie : je débute la lecture en colonne 1 de la variable « nom » avec le format « $CHAR12. » . Dans le langage SAS, le format « $CHAR*long* » est relatif à la lecture de chaînes alphanumériques d'une longueur égale à l*ong*.

```
DATA madata ;
INPUT
@1 nom $CHAR12.
@14 sexe $1.
@15 t1 2.
@19 t2 2.
@23 t3 2.
@27 t4 2.
;
LINES ;
Dupont Alain M 2   2   2   1
Dumont Titi  F 2   1   2   1
Dupond Isa   F 1   3   1   2
Sébastien    M 4   4   3   3
;
PROC PRINT DATA = madata ;
TITLE ' impression des donnees lues ';
RUN ;
```

LABEL

L'ordre LABEL permet d'affecter des noms aux variables. Dans ce qui précède, écrivez, juste après DATA madata (première ligne) :

```
LABEL t1='Mathématiques'
t2='Physique' t3='Chimie' t4='Economie;
```

PROC CONTENTS

Enchaînez en écrivant, à la fin de l'étape DATA (donc, dans une nouvelle étape), la procédure CONTENTS qui va éditer le contenu détaillé de la table *madata*.

```
PROC CONTENTS DATA=madata;
RUN;
```

PROC UNIVARIATE

```
PROC UNIVARIATE DATA=madata ;
RUN ;
```

Pour chacune des variables quantitatives de la table « madata », la procédure UNIVARIATE va éditer des statistiques générales telles que moyenne, écart-type et variance.

Le format yymmdd

Quand le format **yymmdd** lit le chiffre 990915, il entend l'année 99, le mois 09, le jour 15, c'est-à-dire yy, comme year, mm, comme month et dd, comme day. Le chiffre qui est lu avec ce format provoque - conventionnellement - le calcul du nombre de jours qui le sépare du 1er janvier 1960.

```
DATA madata ;
INPUT (datedeb datefin) (2*YYMMDD.);
ecoules=datefin-datedeb;
LINES ;
600101601231
620101621231
;
PROC PRINT DATA=madata;
RUN ;
```

Lecture à raison de plus d'une ligne par observation et plus d'une observation par ligne

Dans une étape DATA, avec l'ordre

```
DATA toto ;
INPUT (a b c d e f) (3*2. / 3*2.) ;
```

L'ordre INPUT lit les variables a, b, et c sur une première ligne, puis les variables d, e et f sur la ligne suivante : l'observation se lit sur deux enregistrements. Inversement, grâce au symbole « @@ » on peut lire plusieurs observations sur un même enregistrement. Le format @@ implique une lecture sans saut de ligne à chaque observation. Par exemple, le programme suivant lit 7 observations depuis une seule ligne :

```
DATA toto ;
INPUT entier @@ ;
LINES ;
1 2 3 3 4 2 1
;
PROC PRINT ;
RUN ;
```

SET

Soit la DATA toto1 créée comme suit :

```
DATA toto1 ;
X=1 ; y=2 ; z=3 ; RUN ;
```

Si nous écrivons le programme ci-contre, la DATA SAS toto2 est alors composée avec la DATA SAS toto1 à laquelle la variable « t » a été ajoutée avec t=x+y+z .

```
DATA toto2 ;
SET toto1 ;
t=x+y+z ;
RUN ;
```

SET *liste de DATA ;*

Plus généralement, on pourra écrire :

```
DATA madata ;
SET liste ;
```

où *liste* est la liste des noms des tableaux (DATA) que nous désirons mettre dans la DATA madata.

RENAME

Dans une étape DATA, on peut changer le nom d'une variable en écrivant un ordre de la forme :

RENAME *varold=varnew* ;

KEEP et DROP

On peut vouloir, qu'à la fin de l'étape DATA, ne soient gardées qu'un certain nombre de variables ou éjecter un certain nombre de variables. On peut le faire en écrivant, dans l'étape DATA, un ordre de l'une des deux formes suivantes :

KEEP *liste des variables à garder* ;
DROP *liste des variables à éjecter* ;

Concaténation de deux DATA

Dans l'encadré qui suit, nous créons la DATA « data1 » (2 observations), puis la DATA « data2 » (2 observations). A la 3ème étape, nous créons la DATA « tout » par concaténation des DATA data1 et DATA2. Ainsi la DATA « tout » contient 4 observations. En soumettant le programme de l'encadré qui suit, les choses seront encore plus claires.

```
DATA data1; * Première étape ;
INPUT (var1 var2 var3) (3*2.);
LINES ;
 1 2 1
 2 3 1
;

DATA data2 ;* Deuxième étape ;
INPUT (var1 var2 var3) (3*2.) ;
LINES;
 2 3 4
 1 1 1
;

DATA tout ; * Troisième étape;
SET data1 data2;
PROC PRINT;
TITLE " Impression des données de deux DATA";RUN;
```

Valeurs manquantes

Le programme de l'encadré qui suit ne diffère du précédent que par le nom de la deuxième variable de la DATA data2 (v2 au lieu de var2). Soumettez – le et commentez les résultats.

```
DATA data1; * Première étape ;
INPUT (var1 var2 var3) (3*2.);
LINES ;
 1 2 1
 2 3 1
;

DATA data2 ;* Deuxième étape ;
INPUT (var1 v2 var3) (3*2.) ;
LINES;
 2 3 4
 1 1 1
;

DATA tout ; * Troisième étape;
SET data1 data2;
PROC PRINT;
TITLE " Impression des données de deux DATA";RUN;
```

Calculs usuels à l'intérieur d'une étape DATA

Soumettez le programme suivant et vérifiez que SUM calcule des sommes, MEAN des moyennes, VAR des variances, STD l'écart type, et CV le coefficient de variation.

```
DATA data1; * Première étape ;
INPUT (var1 var2 var3) (3*2.);
LINES ;
 1 2 1
 2 3 1
;
DATA data2 ;* Deuxième étape ;
SET data1;
Somme = SUM (OF var1-var3);
Moyenne = MEAN (OF var1-var3);
Ecart = STD (OF var1-var3);
Varie = VAR (OF var1-var3);
Coevar = CV (OF var1-var3);
PROC PRINT;
RUN;
```

LAG1, LAG2, LAG3, ... ,

Dans une étape DATA, les fonctions LAG1, LAG2, etc. … se réfèrent aux valeurs antérieures d'une variable. Soumettre le programme qui suit permet de le vérifier.

```
DATA data1;
INPUT t1 t2 t3;
LINES;
1 2 3
2 1 1
1 4 9
;
DATA data2;
SET data1;
tacavan1 = LAG1 (t1);
tacavan2 = LAG1 (t2);
tactac=LAG2(t1);
PROC PRINT DATA=data2;
TITLE 'Utilisation de la fonction LAG';
RUN ;
```

RETAIN *variable*

Dans une étrape DATA, l'ordre RETAIN permet de conserver la valeur d'une variable à travers les enregistrements. En soumettant le programme de l'encadré qui suit, vous verrez que le RETAIN a permis de calculer des cumuls.

```
DATA madata ;  RETAIN totocum 0 ;
INPUT toto ;
totocum = totocum+toto;
LINES;
1
2
3
4
;
PROC PRINT ;
RUN;
```

Décomposition d'une DATA

On peut décomposer les observations d'une DATA en deux ou plusieurs DATA en se référant à la valeur prise par une variable. Par exemple, on peut décider d'écrire l'observation dans la DATA "masc" ou dans la DATA "fem" selon que la valeur de la variable "sexe" vaut "M" ou "F". Pour réaliser ce travail, il faut prévoir le nom des deux DATA dès le début de l'étape. Le programme de l'encadré suivant l'illustre. Soumettez-le.

```
DATA masc fem ;
INPUT @1 nom $char12.
      @14 sexe $1.
      @15 t1 2.
      @19 t2 2.
      @23 t3 2.


IF sexe = 'M' THEN OUTPUT masc ;
IF sexe = 'F' THEN OUTPUT fem;
LINES ;
Dupont Alain M 2   2   2
Dupont Isa   F 1   3   2
Paulette     F 1   3   3
Dupond       M 2   1   1
Simone       F 1   2   3
Sebastien    M 4   4   3
;
PROC PRINT DATA=fem ;TITLE ' fichier feminin ' ;
PROC PRINT DATA = masc ;TITLE 'fichier masculin';
RUN ;
```

Section 2 : Les références extérieures

CES EXERCICES SUPPOSENT QUE VOUS AVEZ CHARGE LES FICHIERS SUPPORT DU COURS. SI VOUS NE L'AVEZ PAS ENCORE FAIT, REPORTEZ-VOUS A L'ANNEXE 1.

L'étape DATA invoque souvent des références extérieures, c'est-à-dire, la lecture d'un fichier précédemment créé ou l'utilisation d'un programme qui se trouve stocké dans une bibliothèque de programmes ou encore l'utilisation d'un membre d'une librairie de données extérieures. Les exercices proposés dans ce paragraphe tendent à familiariser le stagiaire avec l'utilisation de références extérieures tout en introduisant de nouveaux ordres.

INFILE

Dans une étape DATA, l'ordre INFILE définit le fichier sur lequel porte la lecture. Par exemple, le programme suivant lit des données sur le fichier UNIX 'ficstag/emploi' et les imprime.

```
DATA madata ;
INFILE 'ficstag/emploi' ;
INPUT an taille secteur hommes femmes salaires;
PROC PRINT ; RUN ;
```

FILE

Alors que INFILE annonce le fichier à lire, FILE annonce le fichier extérieur sur lequel nous écrivons. Par exemple, le programme suivant fait une copie partielle de 'fictag/emploi' sur 'ficstag/extrait' en retenant les variables « an » et « salaires ».

```
DATA madata ;
INFILE 'ficstag/emploi' ;
INPUT an taille secteur hommes femmes ;
FILE 'ficstag/extrait' ; PUT an salaires ;
PROC PRINT ; RUN ;
```

DO ; ... END ;

Dans une étape DATA, entre un DO et un END, on peut intégrer plusieurs ordres. Le programme suivant sélectionne de 'ficstag/emploi' l'année 1982 par exemple. Puis il intègre entre le DO ; et le END ; deux ordres relatifs à l'année 1982.

```
DATA madata ;
INFILE 'ficstag/emploi';
INPUT annee taille secteur hommes femmes ;
IF annee=82 THEN DO ;
FILE 'ficstag/truc' ; PUT annee taille secteur hommes femmes ;
END ;
PROC PRINT ; RUN ;
```

FILENAME

FILENAME établit une correspondance entre le nom physique d'un fichier, comme par exemple, 'fictag/emploi' et un nom logique valable pour la durée de la session. Par exemple, vous pouvez commencer votre programme SAS en écrivant dès avant l'étape DATA :

```
FILENAME toto 'ficstag/emploi' ;
```

Cette instruction dit que l'unité physique ficstag/emploi va être désignée au cours de la session par *toto*. Le programme suivant illustre l'utilisation du FILENAME. Soumettez-le.

```
FILENAME fifi 'ficstag/emploi'  ;
FILENAME sortie ' ficstag/truc' ;
DATA _NULL_; INFILE  fifi ;
INPUT annee taille secteur hommes femmes salaires depenses ;
IF annee = 82 THEN DO ;
     FILE sortie ;
PUT annee taille secteur hommes femmes depenses salaires;
END;
RUN;
```

Dans ce qui précède **FILE sortie** a annoncé l'écriture sur le fichier dont le nom logique est « sortie » ; c'est-à-dire, en vertu du FILENAME ci-dessus, sur le fichier 'ficstag/truc'. Le « PUT » de l'instruction qui suit ordonne l'écriture sur « sortie ».

De l'utilité du FILENAME

Sachant que 'ficstag/emploi' est une syntaxe UNIX, quand vous utiliserez 'ficstag/emploi' plusieurs fois dans le même programme (disons 10 fois) et que vous voudrez déménager votre programme vers un micro-ordinateur fonctionnant sous DOS, il faudra alors remplacer la syntaxe UNIX (ici, 'ficstag/emploi') par la syntaxe DOS (ici, 'c :\ficstag\emploi'). Si, au contraire, vous avez dit « fifi » au lieu de « 'ficstag/emploi' », vous n'aurez qu'à modifier le FILENAME.

LIBNAME

LIBNAME est utilisé pour associer le nom physique d'un répertoire permanent à un nom logique valable pendant une session SAS. Soit « libdata » le nom UNIX du répertoire préalablement crée et supposons que nous voulions nous référer à cette librairie, au cours d'une session SAS, via l'appellation « malib ». On écrira alors :

LIBNAME malib 'libdata' ;

Des cotes ont été nécessaires autour de 'libdata' pour dire que c'est un nom étranger à SAS, car c'est un nom reconnu par le système d'exploitation (UNIX sur les ordinateurs de l'université de Paris X – Nanterre), physiquement gravé sur une unité périphérique. Maintenant, pour créer une librairie du nom de 'libdata', faire comme suit :

- Créer le répertoire UNIX « libdata »
- Revenir à SAS.
- Maintenant, on va mettre dans « libdata » des bases SAS. Pour ce faire, on affectera au nom PHYSIQUE libdata (permanent) un nom LOGIQUE (valable pour la session) ; par exemple « toto »
- Soumettez le programme suivant.

```
LIBNAME toto 'libdata' ;
DATA toto.essai ;
x=1;y=2;z=3;
RUN ;
```

- Sortez de SAS vers UNIX
- Listez les membres du répertoire « libdata » en observant que le membre « essai.ssd01 » a été créé. L'extension « ssd01 » vous dit que c'est un fichier de type SAS (source system data).

Toto.essai est désormais une base permanente. J'éteins, je pars en vacances trois mois. Je reviens des vacances, j'écris le programme SAS suivant :

```
LIBNAME truc 'libdata' ;
PROC PRINT DATA= truc.essai; RUN;
```

et je retrouve mes données imprimées sans en préciser le format d'écriture.

FILENAME ET LIBNAME ENTRE UNIX et SAS

Qu'il s'agisse du FILENAME ou du LIBNAME, il faudra toujours faire attention à la racine sous laquelle vous avez invoqué SAS. Pour l'utilisateur Dupont de l'Université de Paris-X Nanterre par exemple qui a crée sa librairie de données sous le nom physique libdata et qui a invoqué SAS sous la racine :

home/users/export/sceco/dupont

LIBNAME malib 'libdata' ; entend qu'il possède le répertoire UNIX libdata localisé comme suit :

home/users/export/sceco/dupont/libdata

Si le même Dupont avait invoqué SAS sous la racine :

home/users/export/sceco/dupont/coursas

LIBNAME malib 'libdata' ; aurait entendu :

home/users/export/sceco/dupont/coursas/libdata

Création de bases dans la librairie « libdata »

Ce programme lit un fichier d'indices internationaux (INPUT...) et stocke le résultat de la lecture dans la librairie « libdata ». Soumettez-le.

```
LIBNAME malib  'libdata' ;

DATA malib.places;
INFILE  'ficstag/places' ;
INPUT (mois cac dj ftm can net bel ger swi ita spa nik syd khg)
(14*4.) ;
PROC PRINT; RUN ;
```

Faire de même en créant dans la librairie SAS « libdata », les membres suivants :

- emploi : en lisant dans 'ficstag/emploi', les variables « annee taille secteur hommes femmes salaires depenses ». L'ordre INPUT s'y fait en format libre : INPUT annee taille secteur …
- structur : en lisant dans le fichier 'ficstag/structur », les variables « secteur onq oq emp tam ic sonq soq semp stam sic) . L'ordre INPUT s'y fait en format libre : INPUT secteur onq oq emp tam ....
- money : en lisant dans le fichier 'ficstag/money'. Nous lirons une série de 9 entiers collés, chacun d'eux s'étendant sur 4 colonnes : INPUT (mois .....)(9*4.) ;

Section 3 : La programmation avancée

Nous allons aborder des exercices de programmation un peu moins courts que les précédents mais tout aussi faciles.

PROC SORT

En programmation, les tris sont souvent nécessaires. La procédure SORT de SAS les réalise. Par exemple, le programme suivant, trie la table malib.emploi par valeurs ascendantes du taux de féminisation des strates étudiées.

```
DATA jetrie ;  SET malib.emplois ;
tofemmes = 100 * (femmes / hommes ) ;
PROC SORT DATA = jetrie OUT = fini ; BY tofemmes ; RUN ;
```

Pour un tri par valeurs descendantes de la variable « tofemmes », ajouter l'option « DESCENDING » comme suit :

```
PROC SORT ; BY DESCENDING tofemmes ;
```

On peut aussi trier une DATA par référence à un plus grand nombre de critères. Par exemple, le programme suivant trie « malib.emploi »

- d'abord par la variable « année »,
- ensuite, et pour chaque année, les observations sont triées par le secteur,
- Ensuite, et pour chaque strate année*secteur, les observations sont triées par référence à l'ordre descendant des classes de tailles.

```
PROC SORT DATA=malib.emploi OUT=sortie ;
BY annee secteur DESCENDING taille ;
RUN ;
```

MERGE

Vous disposez d'une librairie SAS où le membre « places » contient le prix de treize indicateurs financiers internationaux alors que le membre « money » contient le prix de huit monnaies exprimées en francs français. Tous les deux contenant par ailleurs, la même variable « mois ». L'instruction MERGE va permettre de créer un troisième fichier où seront mises côte à côte les variables de ces deux fichiers. Si dans le fichier ainsi crée, pour un même valeur de la variable « mois » nous avons les deux types de données, nous pourrons écrire l'instruction :

cacdol=cac/usd ;

cacdol est ainsi le prix du CAC en dollars américains.

Si vous écrivez le programme suivant :

```
LIBNAME malib  'libdata' ;
DATA fusion ; MERGE malib.places malib.money ; RUN ;
```

vous mettrez, dans la table « fusion », la nème observation de malib.places à côté de la nème observation de malib.money et ce, compte non tenu de la variable « mois » qui aurait dû être votre critère de fusion. C'est une fusion un à un (*one-to-one merging*). Comme les deux tables (malib.places et malib.money) n'ont pas le même nombre d'observations, vous aurez aussi (dans la table fusion) des valeurs manquantes (désignées par SAS avec le symbole '.'). Or, nous voulons une fusion telle que pour l'observation où mois = 9802 (février 1998) nous ayons côte à côte les deux observations de malib.places et de malib.money pour lesquelles mois=9802, ce que la syntaxe du langage SAS autorise le « *match merging* » comme suit :

```
DATA fusion ;
MERGE malib.places malib.money ;
BY mois ;
```

Dans le cas général, la fusion ne va cependant pas s'effectuer correctement si les deux tables ne sont pas triées par ordre croissant de « mois » (critère de la fusion) et le programme final devra ainsi être :

```
PROC SORT DATA=malib.places OUT=x ; BY mois ;
PROC SORT DATA=malib.money OUT=y ; BY mois ;
DATA fusion ; MERGE x y ; BY mois ; RUN ;
```

Il est enfin possible d'introduire une variable logique comme option de la DATA (*data set option)* comme suit :

```
DATA fusion ; MERGE malib.places (IN=in1) malib.money(IN=in2) ;
BY mois ;
IF in1 AND in2 ;
RUN ;
```

« IF in1 AND in2 » équivaut à « IF in1 AND in2 THEN OUTPUT ». C'est un OUTPUT implicite. Quant à in1 et in2, ce sont des variables logiques : le OUTPUT (l'écriture sur la table « fusion ») ne va s'effectuer que si les conditions in1 et in2 sont simultanément réalisées, c'est-à-dire, que si la modalité de la variable mois (la variable critère de la fusion) a été aussi bien trouvée dans la table malib.places que dans la table malib.money.

Applications pour se familiariser avec le MERGE

- Remplacer « IF in1 AND in2 » par « IF in1 AND NOT in2 » , la DATA fusion contiendra ainsi les observations pour lesquelles la variable « mois » manque dans malib.money.
- Créer la table « madata » qui contient seulement :
    - la variable « mois »
    - cacdol=cac/usd ;
    - ftmdol= la variable FTM de malib.places exprimés en dollars US.
    - gerdol = la variable GER de malib.places en dollars US.
    - spadol = SPA exprimé en dollars US.
    - nikdol = NIK exprimé en dollars US.
- Faire un PROC UNIVARIATE et/ou un PROC CONTENTS de la DATA fusion.

RETAIN *liste de variables* **(rappel)**

RETAIN a b c 0 ;

Dans une étape DATA, « RETAIN a b c 0 ; » a pour effet de garder la valeur des variables a, b et c disponibles à travers les observations, leur valeur initiale étant égale à zéro. Son utilisation est donc identique à celle vue précédemment pour une seule variable. Dans le programme qui suit, par exemple, le RETAIN est utilisé pour calculer les cumuls, à travers les enregistrements, des variables cac, dj et ftm :

```
DATA madata ; SET malib.places ;
RETAIN caccun djcum ftmcum 0 ;
caccum=caccum+cac; djcum=djcum+dj; ftmcum=ftmcum+ftm;
PROC PRINT;
RUN;
```

ARRAY

Dans une étape DATA, ARRAY définit un vecteur. Ainsi, le premier des deux ARRAY qui suivent rend are(1)=a ; are(2)=b , are(3)=c ; are(4) = d ; etc.... et DIM(are) = 10 . DIM(are) étant la dimension du vecteur que nous avons appelé are. Le second ARRAY rend are(1)=var1, are(2)=var2, ..., are(99)=var99, are(100)=var100 , et DIM(are)=100.

```
/* Je suis à l'intérieur d'une étape DATA */
ARRAY are(*) a b c d e f g h i j ;
ARRAY are(*) var1-var100 ;
```

DO ; ... ; END ;

```
DATA newdata ; SET malib.places ;
RETAIN cumul1-cumul5 0 ;
ARRAY arcumul(*) cumul1-cumul5 ;
ARRAY arvect(*)  cac dj ftm can net  ;

    DO indice = 1 TO DIM(arcumul);
    arcumul(indice)=arcumul(indice)+arvect(indice);
   END;

PROC PRINT; ID date ; TITLE " Valeurs et valeurs cumulées " ;
RUN;
```

Dans l'instruction :

```
arcumul(indice)=arcumul(indice)+arvect(indice);
```

l'indice varie de 1 à 5. La boucle DO nous permet ainsi d'éviter l'écriture des 5 instructions suivantes :

```
arcumul(1)=arcumul(1)+arvec(1) ;
….
arcumul(5)=arcumul(5)+arvec(5) ;
```

SUBSTR

Dans une étape DATA, l'ordre SUBSTR (substring) est notamment utile quand nous avons besoin de regarder la chaîne de caractère par laquelle débute la ligne afin de reconnaître le type d'enregistrement. C'est le cas dans le fichier chargé ficstag/stat dont voici un extrait :

Extrait du fichier chargé dans ficstag/stat

```
Date      1931
Fenetres  GALLUP ROOSEVELT

   George Horace Gallup (1901-1984), 30 ans en 1931

   Journaliste et statisticien américain qui fonda le premier institut
   de sondage pour l'étude de l'opinion publique américaine......
   Les travaux de Gallup témoignent du caractère récent de la théorie des
   sondages ,  n'ayant été stimulée qu'en 1936, lorsque des sondages
   donnaient  gagnant  aux présidentielles américaines, Alf Landon contre
   Franklin D. Roosevelt.
```

Ce fichier contient types d'enregistrements (de lignes)

- Ceux qui commencent par la chaîne de caractères « Date »,
- Ceux qui commencent par « Fenetres »,
- Ceux qui commencent par deux espaces blancs.

La fonction SUBSTR va nous servir à reconnaître le type d'enregistrement et à créer trois DATA séparées les unes des autres. Cette séparation en trois fichiers distincts va rendre la gestion et l'analyse de l'événement formulé verbalement beaucoup plus commode. Le déroulement logique du programme de création des trois fichiers sera la suivant :

- Si la ligne commence par « Da » l'enregistrement donne la date
- Si la ligne commence par « Fe » l'enregistrement donne les mots-clés
- Si la ligne commence par des blancs (' ') l'enregistrement est le texte proprement dit.

```
DATA date fenetres ecrits ;  RETAIN numpar 0;
INFILE  'ficstag/stat' PAD LRECL=75;
INPUT ligne $char75. ;
debut = SUBSTR (ligne,1,2);
   IF debut='Da' THEN DO ;numpar=numpar+1; OUTPUT DATE ; END;
   IF debut='Fe' THEN OUTPUT Fenetres;
   IF debut='  '  THEN OUTPUT Ecrits ;
PROC PRINT; VAR ligne ;
RUN ;
```

Observations :

- Après lecture de la ligne de 75 caractères (INPUT ligne $CHAR75.) , le programme écrit (fait son OUTPUT), selon le cas sur l'une des trois tables SAS en cours de création (définies dans l'instruction DATA)
- Les précisions PAD et LRECL=75 permettent de lire la ligne de 75 caractères telle qu'elle apparaît à l'écran lorsque nous l'éditons.

Partie = SUBSTR (Mot, 3, 4) ;

« Partie » est la partie de « Mot » qui commence par la 3ème lettre de « Mot » et qui s'étend sur 4 lettres. De même dans ce qui précède, « debut » est la sous-chaîne de caractères qui représente les deux premières lettres de la variable « ligne ».

Section 4 : Les procédures générales

Parmi les procédures les plus utilisées, PROC SUMMARY et PROC FREQ sont incontournables. SUMMARY permet d'agréger les observations autour d'une variable discrète ou qualitative et FREQ permettant d'établir des tableaux de contingence, éventuellement pondérés. Mais il est bon aussi, pour des questions de visualisation, de savoir utiliser PROC PLOT pour les représentations graphiques, PROC FORMAT pour recoder des variables discrètes, et PROC CHART pour dessiner histogrammes et camemberts.

PROC SUMMARY

PROC SUMMARY fait des calculs de moyenne, somme, écart-type, variance, coefficient de variation, etc. Commencez par soumettre le programme :

```
DATA x ;
INPUT  section $ t1 t2 t3 t4 ;
LINES ;
a 1 2 1 2
b 2 2 1 1
a 1 1 1 4
b 2 2 2 1
;
PROC SUMMARY DATA=x ; CLASS section ;
VAR t1-t4 ;
OUTPUT OUT=y MEAN=mt1-mt4 ;
PROC PRINT ; RUN ;
```

Dans le programme ci-avant, le mot-clé MEAN permet de calculer les moyennes des variables t1 à t4 pour chaque catégorie de la variable « section » (deux modalités : a et b). La variable de catégorisation est annoncée par le mot-clé CLASS .

OUTPUT OUT=y a pour effet de nommer « y » la DATA SAS de travail où nous stockons la sortie de PROC SUMMARY où la moyenne a été calculée.

MEAN=mt1-mt4 a pour effet d'appeler mt1, mt2, mt3, et mt4 les moyennes calculées de t1 à t4. De la même manière que MEAN a été utilisé, les mots clés suivants peuvent aussi être utilisés :

- SUM = pour le calcul des sommes
- VAR = pour le calcul des variances,
- STD = pour le calcul de l'écart-type,
- CV = pour le coefficient de variation
- STDERR= pour l'erreur standard des moyennes (intervalle de confiance)

PROC SUMMARY
La variable _TYPE_

Le programme :

```
PROC SUMMARY DATA=malib.emploi ; VAR hommes femmes ;
CLASS annee taille secteur ; OUTPUT OUT=ficsor  SUM= somh somf;
```

crée une DATA SAS ficsor qui contient la variable _TYPE_ . TYPE_ se lit comme suit :

_ TYPE_	annee	taille	secteur
0	. (1)	. (2)	. (3)
1	.	.	détail des secteurs
2	.	détail des tailles	.
3	.	détail des tailles	détail des secteurs
4	détail des années	.	.
5	détail des années	.	détail des secteurs
6	détail des années	détail des tailles	.
7	détail des années	détail des tailles	détail des secteurs

(1) Toutes années confondues (2) Toutes tailles confondues (3) Tous secteurs confondus.

Pour comprendre la logique de la variable _TYPE_ , écrire en binaire les entiers naturels allant de 1 à 7 en soumettant le programme suivant :

```
DATA  toto;
INPUT entier @@;
binaire = PUT (entier, BINARY3.) ;
LINES;
1 2 3 4 5 6 7
;
PROC PRINT; RUN;
```

PROC FREQ

PROC FREQ calcule des fréquences, le chi2 des cases, la valeur attendue, etc. En écrivant :

```
LIBNAME malib 'libdata' ;
PROC FREQ DATA=malib.emploi ;
TABLES annee ; RUN ;
```

Nous obtenons la distribution des fréquences de la variable « annee » dans la base malib de la librairie libdata alors qu'avec

```
PROC FREQ DATA=malib.emploi ;
TABLES annee*taille ; RUN ;
```

nous obtenons la fréquence des années par les tailles. Si on veut des fréquences pondérées, il faut ajouter l'ordre WEIGHT. Par exemple :

```
PROC FREQ DATA=malib.emploi ;
TABLES annee*taille ; WEIGHT salaires ; RUN ;
```

Cas général :

PROC FREQ;TABLES secteur* taille / NOROW NOCOL NOPERCENT
EXPECTED CELLCHI2 DEVIATION ;

quelques options sont possibles avec FREQ :

NOPRINT = pas d'impression
OUT = = nom SAS du tableau de sortie
NOROW = pas de pourcentage sur les lignes
NOCOL = pas de pourcentage sur les colonnes
NOPERCENT = pas de %age relativement au total
EXPECTED = valeur attendue
CELLCHI2 = contribution de la case au chi2 total
DEVIATION = déviation par rapport à EXPECTED

PROC FORMAT

Une des fonctions de la procédure FORMAT est de faire des recodages. On sait que la variable « secteur » du membre « emploi » de la librairie « libdata » prend des valeurs qui varient entre 1 et 38 (voir annexe 2) . Le programme ci-après crée une fonction « recodage » qui regroupe les secteurs de l'APE 40B (Cf. anexe n°3) en quatre catégories qui s'appellent respectivement « primaire », « secondaire », « BTP » et « tertiaire ».

```
LIBNAME malib 'libdata' ;
PROC FORMAT ;
VALUE recodage
1-3='primaire'
4-21='secondaire'
24='BTP'
OTHER='tertiaire';
DATA x;SET MALIB.emploi;
sect=PUT (secteur,recodage.);
PROC PRINT ; RUN ;
```

LA POLYSEMIE DU PUT

Dans SAS, le mot-clé PUT est polysémique. En effet dans ce qui précède, l'acception du PUT dans sect=PUT(secteur,recodage.) est l'ordre par lequel la nouvelle variable sect est l'image de la variable secteur à travers le format "recodage" préalablement défini par un PROC FORMAT alors que dans le programme suivant

```
DATA madata ;
INFILE 'ficstag/emploi' ;
INPUT an taille secteur hommes femmes ;
FILE 'ficstag/extrait' ; PUT an salaires ;
PROC PRINT ; RUN ;
```

l'acception du PUT (dans PUT an salaires) est un ordre d'écriture sur l'unité physique ficstag/emploi.

Utilisation du PROC FORMAT pour l'écriture de libéllés

Lisons les données du fichier fictag/banques

```
DATA madata ;
INFILE 'ficstag/banque';
INPUT ident $ 4. a78-a82; [1]
```

[1] Ces données proviennent d'un article publié par M.-A. Lévy dans Economie et Statistique n° 234. Juillet-août 1990.

Nous pouvons maintenant rédiger un PROC FORMAT pour affecter à chacune des modalités prises par la variable ident son sens en toutes lettres comme suit :

```
PROC FORMAT ;
VALUE $for
'cctf' = 'credit ct francs '
/* Pour compléter, Cf. annexe 2 ficstag/banque */
'oper' = 'operations interbancaires'...
;
```

Dans ce qui précède, le FORMAT $for a affecté des appellations à une série de modalités. Si on écrit :

```
PROC PRINT DATA=madata; FORMAT  nom $for. ;
```

Les appellations définies par le FORMAT $for seront considérées.

PROC STANDARD

PROC STANDARD permet notamment de régler la moyenne et l'écart-type des variables d'une DATA SAS. Considérons le membre « places » de la librairie « libdata » et calculons le CAC et le Dow Jones de manière à ce qu'ils aient tous les deux une moyenne nulle et un écart-type égal à l'unité. La procédure STANDARD permet de procéder aux calculs requis en écrivant :

```
PROC STANDARD DATA=malib.places MEAN=0 STD=1 OUT=masortie ;
VAR cac dj ;RUN ;
```

Dans ce qui précède OUT=masortie a permis de donner le nom « masortie » à la DATA SAS en sortie qui contient les valeurs du CAC et du Dow Jones centrées réduites.

PROC PLOT

PROC PLOT permet d'obtenir des représentations graphiques. On peut enchaîner avec le PROC STANDARD qui précède, en demandant une représentation graphique simultanée des variables « cac » et « dj » par l'ordre temporel (variable « ordr »=_N_)

```
DATA masortie ; SET masortie ; ordre=_N_ ;
PROC PLOT ; PLOT cac*ordr='.' dj*ordr='*'/OVERLAY ;RUN ;
```

Dans ce qui précède, l'option OVERLAY signifie que nous demandons une représentation simultanée sur un seul graphique des variables cac et dj.

PROC TRANSPOSE

PROC TRANSPOSE transpose une DATA : les lignes deviennent des colonnes et inversement. Créons le membre « banques » de la librairie de données 'libdata' par référence au fichier ficstag/banque.

```
LIBNAME malib 'libdata' ;
DATA malib.banques ;
INFILE 'ficstag/banque;
INPUT nom $ 4. a78-a82;
RUN;
```

Faire les trois essais suivants en imprimant, à chaque essai, la table SAS en sortie du PROC TRANSPOSE, que nous appelons « sortie ».

- PROC TRANSPOSE DATA=malib.banques OUT=sortie ;
- PROC TRANSPOSE DATA=malib.banques OUT=sortie NAME=ident ;
- PROC TRANSPOSE DATA=malib.banques OUT=sortie NAME=ident ; ID nom ;

Dans le premier de ces trois essais, on observe que PROC TRANSPOSE a crée la variable _NAME_ pour contenir le nom des variables du fichier original avant transposition. Le second a remplacé _NAME_ par « ident ». Dans les deux premiers cas, les variables du fichier transposé portent des noms de la forme COL1, COL2, etc. alors qu'en ajoutant

ID nom ;

dans le troisième cas, les nom des variables du fichier transposé correspondent aux modalités de la variable « nom ».

PROC CORR

PROC CORR calcule les corrélations entre variables d'une DATA.

```
PROC CORR data = malib.places OUTP=datapear OUTS=dataspir NOPRINT;
VAR dj  ; WITH cac dj ftm ;
PROC PRINT DATA = dataspir ; TITLE 'data de Spearman ';
PROC PRINT DATA = datapear ; TITLE 'data de Pearson  ';
RUN;
```

Dans ce qui précède,

- DATA=malib.places signifie que les calculs sont à effectuer depuis la DATA SAS permanente malib.sas.
- OUTP=datapear signifie que nous allons récupérer les résultats des calculs - pour ce qui est des coefficients de Pearson - dans la DATA SAS datapear.
- OUTS = dataspir signifie que nous allons récupérer les résultats des calculs - pour ce qui est des coefficients de Spearman - dans la DATA SAS dataspir.
- NOPRINT signifie que nous n'imprimons pas les résultats.

PROC CHART

La procédure CHART permet d'imprimer des histogrammes et des camemberts. Dans l'exercice qui suit, si l'ordre PIE ne donne pas de camembert, consulter le compte-rendu du déroulement logique et régler les options LS et PS (Line size et Page Size)

OPTIONS LS=... PS=... ;

Pour être assuré d'avoir un camembert, choisir LS=200 et PS=130. (LS et PS sont des abréviations de Line Size et de Page Size)

```
DATA madata ;
INFILE 'ficstag/emploi' ;
INPUT annee taille secteur hommes femmes salaires depenses ;
PROC CHART ; STAR annee / DISCRETE ;
TITLE ' Repartition de l ''annee ' ;
PROC CHART DATA=madata; VBAR annee
  / SUMVAR=depenses TYPE=SUM DISCRETE GROUP=taille ;
TITLE ' Depenses representees par barre verticale Vbar ' ;
PROC CHART DATA=madata; PIE  annee  / SUMVAR=depenses TYPE=MEAN  ;
TITLE ' Les depenses moyennes en forme de camembert ' ;
```

Section 5 : Les procédures statistiques

SAS met à la disposition de l'utilisateur de nombreuses procédures statistiques. L'essentiel est ici de rentrer dans l'esprit de la programmation qui devrait permettre à l'utilisateur de circuler à travers le langage, comme à travers l'enchaînement des procédures car souvent une application particulière fait appel à plus d'une procédure.

PROC DISCRIM
Analyse discriminante

Considérons le programme suivant :

```
DATA toto ;
INPUT proprio $ revenu ;
LINES ;
oui  9999
oui  9813
non 5500
non 4000
oui 6900
non 7000
oui 10920
non 4598 ;
```

Dans ce qui précède, « proprio » est une variable qualitative qui a deux modalités « oui » et « non ». PROC DISCRIM peut distinguer les deux modalités par référence à la variable « revenu ». « proprio » est ainsi une variable « cible » dans la discrimination. PROC DISCRIM regroupe les observations par modalités et donne des probabilités *a posteriori* de « proprio ». Pour discriminer les observations par référence à la variable « proprio », on écrira :

```
PROC DISCRIM DATA=toto OUTCROSS=calibre ;
CLASS proprio ; VAR revenu ;
PROC PRINT DATA=calibre ;
TITLE " Calibrage de la variable proprio par la variable revenu " ;
PROC FREQ; TABLES proprio*_INTO_;
TITLE " Variable proprio par probabilité postérieure" ;RUN;
```

LA VARIABLE _INTO_ DANS PROC DISCRIM

-INTO_ est une variable créée par PROC DISCRIM dans la DATA SAS « calibre ». Comme nombre de variables créées par SAS, elle est de la forme _xxx_ . _INTO_ Dantable SAS « calibre », _INTO_ vaudra « oui » quand la probabilité a posteriori d'être propriétaire est supérieure à celle de ne pas l'être. Imprimer la table « calibre » et regarder les probabilités prises par les variables « oui » et « non ».

PROC REG
Régression

Considérons le programme suivant :

```
DATA toto;
INPUT proprio $ revenu age @@  ;
LINES ;
oui  9999  26 oui 9013 26 oui 9999 27  oui 10432 28
oui 9777 25 oui 8888 25 oui 8800 24 oui 10000 28
non  8899  25 non 8013 24 non 8999 25  non  9232 22
non 7000 21 non 8200 22 non 7800 24 non 7100 20 ;
```

A partir de la DATA qui précède, nous pouvons procéder à une estimation de la variable « revenu » par la variable « age » en écrivant tout simplement :

```
PROC REG DATA=toto ;
MODEL revenu=age ;
TITLE 'Estimation du revenu par l'âge' ;
RUN;
```

PROC REG permet aussi de faire des estimations séparées pour chacune des modalités de la variable « proprio ». Mais il faut qu'au préalable, « proprio » soit trié avec un PROC SORT.

On écrira alors :

```
PROC SORT DATA=toto ; BY proprio ;
PROC REG DATA=toto ;
MODEL revenu=age ;
BY proprio ;
TITLE
'Estimations du revenu par l'âge, selon que l'on est
propriétaire ou pas' ;
RUN ;
```

Nous pouvons aussi introduire des contraintes en écrivant :

RESTRICT *contrainte* ;

Par exemple, dans le fichiers des indices internationaux (placés dans la librairie libdata), on peut estimer le CAC par les indices Dow Jones, Financial Times, et Nikkei (codés respectivement dj, ftm et nik) tout en imposant des coefficients de régression relatifs au Financial Times et au Nikkei égaux comme le montre l'exemple à la page suivante.

```
PROC REG DATA=malib.places;
MODEL cac=dj ftm nik ;
RESTRICT ftm=nik ;
RUN ;
```

Régression typologique avec PROC DISCRIM et PROC REG

Dans le programme suivant :

```
DATA toto;
INPUT proprio $ revenu age @@  ;
LINES ;
oui  9999  26 oui 9013 26 oui 9999 27  oui 10432 28
oui 9777 25 oui 8888 25 oui 8800 24 oui 10000 28
non  8899  25 non 8013 24 non 8999 25  non  9232 22
non 7000 21 non 8200 22 non 7800 24 non 7100 20
 ;
PROC DISCRIM OUTCROSS = calibre (KEEP=_INTO-) ;
CLASS proprio ;
```

PROC DISCRIM considère « proprio » comme variable de classification *a priori*.. Grâce à PROC REG, nous pouvons aussi faire des régressions typologiques en utilisant la probabilité de classification *a posteriori* (variable _INTO_ de la DATA en sortie de PROC DISCRIM) :

```
PROC DISCRIM DATA=toto OUTCROSS=sortie ;
CLASS proprio ; VAR revenu age ; RUN ;
PROC SORT DATA=sortie ; BY _INTO_ ;
PROC REG DATA=sortie ; MODEL revenu=age ; BY _INTO_ ;
RUN ;
```

Question :

Qu'est-ce qui se passe si on ne procède pas à un PROC SORT avant de faire la discrimination avec le BY _INTO ? Si vous ne le savez pas, essayez de le faire.

PROC REG
Récupérer les coefficients de la régression dans un DATA SAS

Les coefficients de la régression effectuée par PROC REG peuvent être récupérés dans une DATA SAS définie par l'option OUTEST=*nom* où nom est le nom que l'on désire donner à cette DATA. Dans l'exercice qui suit, cette DATA a été appelée « sorind ».

```
/* Définir l'unité physique malib.place avec LIBNAME */
PROC REG DATA=malib.places OUTEST=sorind;
MODEL cac=dj ftm ger ;
DATA sorind ; SET sorind ;
RENAME dj=coef_dj ftm=coef_ftm ger=coef_ger ;
KEEP INTERCEP  dj  ftm  ger ;
PROC PRINT; TITLE 'Indices de la régression';
```

Dans ce qui précède, nous n'avons gardé des variables de OUTEST que les paramètres de la régression : dj, ftm, ger, et la constante que PROC REG appelle par construction : INTERCEP. Il s'agit maintenant de fusionner ces paramètres avec le fichier original « malib.places ». Cette fusion ne crée pas de problème de confusion puisque dans la DATA « sorind » créée plus haut les coefficients ne portent plus le nom des variables auxquelles ils correspondent, mais des noms différents. Ainsi le coefficient de la variable « dj » s'appelle « coef_dj ». Il est donc aisé de fusionner avec un MERGE sans risque, en écrivant :

```
DATA x ; SET malib.places ; fictif=1;
DATA y ; SET sorind ; fictif=1;
DATA melange ; MERGE x y ; BY fictif ;
```

Pour estimer le CAC par référence aux coefficients calculés, on écrira donc :

```
DATA melange ; SET melange ; DROP fictif ;
cacest = INTERCEP  +  ...*dj +    ...*ftm + . *ger;
PROC PRINT ; TITLE 'CAC calculé avec les indices de la
régression' ; RUN ;
```

Proposition : calculer la valeur estimée de cac (cacest) en comblant les points de suspension et vérifier que la valeur de « cacest » ainsi obtenue est bien la même que celle qu'on aurait obtenue dans la DATA « toto » en faisant OUTPUT OUT=toto P=cacest au niveau du PROC REG.

PROC CORRESP
Analyse des correspondances

Dans le programme suivant, PROC CORRESP réalise une analyse factorielle des correspondances sur les variables définies par l'ordre VAR (c'est-à-dire a78-a82) et les identifie par l'identifiant défini par l'ordre ID (c'est-à-dire : nom).

```
/* Définir avec LIBNAME l'unité logique malib */
PROC CORRESP DATA=malib.banques
DIMENS = 3 OUTC = masortie ;
VAR a78-a82 ;
ID nom ;
PROC CONTENTS DATA=masortie ;
RUN;
```

- DIMENS=3 signifie qu'on calcule trois facteurs
- OUTC=sortie signifie que la table de sortie des résultats de l'analyse va s'appeler « sortie ».
- DIMENS et OUTC sont optionnels.
- ID se rapporte à une variable alphanumérique, faute de quoi (et contrairement à d'autres procédures) PROC CORRESP ne tournera pas.

Application

1° - Sachant que dans la DATA SAS « madata » que vous venez d'obtenir avec « PROC CONTENTS » DIM1 à DIM3 sont les coordonnées sur les axes, faire une représentation graphique des deux premiers axes avec : « PROC PLOT ; PLOT DIM1*DIM2=nom $ nom /VREF=0 HREF=0 ; »

2°- Trier la DATA SAS « masortie » pat _TYPE_ et CONTR1 ;

3° - Imprimer DIM1, CONTR1, SQCOS1 et ID par _TYPE_ (OBS pour observations et VAR pour variables)

4° -Faire : :

- signe=ABS(DIM1)/DIM1)
- Présenter les statistiques de l'axe 1 triées par inertie décroissante précédée du signe qui indique la position de part et d'autre de l'axe.
- Vérifier que les moments d'inertie sont bien nuls (somme MASS*DIMalfa)

PROC CLUSTER et PROC TREE

PROC CLUSTER procède à des classifications ascendantes hiérarchiques. Cette procédure exige que l'on définisse le critère de l'agrégation avec l'option METHOD= . On y dispose notamment des deux méthodes d'agrégation suivantes :

- l'option METHOD=AVERAGE considère que la distance entre deux groupes d'observations est la distance moyenne entre paires d'observations.
- l'option METHOD=CENTROID considère que la distance entre deux groupes d'observations est celle qui sépare les deux centres de gravité des deux groupes.

PROC CLUSTER va souvent de pair avec une procédure de dessin d'un arbre qui s'appelle PROC TREE. Et comme la DATA créée par PROC CLUSTER est conçue de sorte à ce qu'elle puisse fonctionner comme une entrée de PROC TREE, on pourra écrire un programme de la forme :

```
PROC CLUSTER DATA=... METHOD=...;
VAR ... ;
PROC TREE ;
RUN ;
```

Ainsi, on peut faire une PROC CLUSTER sur les trois indices internationaux CAC, Dow Jones et Financial Times avec la méthode CENTROID, en écrivant :

```
PROC CLUSTER DATA=malib.places METHOD=CENTROID ;
VAR cac dj ftm ;
PROC TREE ;
RUN ;
```

Plus généralement, on peut aussi faire un CLUSTER dans l'espace des facteurs issus d'une analyse factorielle des correspondances en écrivant :

```
DATA x ;
SET malib.places ;
LENGTH ident $5. ;
ident=mois ;
PROC CORRESP DATA = x  DIM=3   OUTC=toto NOPRINT ;
VAR cac dj ftm can net ;
ID ident ;
PROC CLUSTER DATA=unedata METHOD=CENTROID ;
VAR DIM1-DIM3; ID ident;
PROC TREE ;
RUN ;
```

Statistiques de PROC CLUSTER

Number of Clusters	------Clusters Joined------			Frequency of New Cluster	Normalized Centroid Distance
11	9007		9010	2	0.042591
10	9006	CL11		3	0.078902
9	9005		9008	2	0.108660
8	9011		9012	2	0.134522
7	CL10		9009	4	0.161067
6	9001		9002	2	0.193301
5	9004	CL9		3	0.291334
4	CL7	CL8		6	0.318406
3	CL6		9003	3	0.424941
2	CL5	CL4		9	0.623542
1	CL3	CL2		12	1.390779

Lire :

Number of Clusters : le numéro d'ordre du lien (11 = le lien le plus fort , ..., 1 = le lien le moins fort)

Clusters Joined : les identifiants qui se joignent. Attention CL11 est l'ensemble des points unis à Number of Clusters n°11

Frequency : nombre total de points liés à ce stade.

Normalized Centroid distance : distance entre les deux points ou agrégats liés au stade concerné par la ligne.

Arbre dessiné par PROC TREE

L'arbre que dessine PROC TREEE est dans la partie inférieure. La partie supérieure ébauche un "dendrogramme "à la française" .

```
                                   (3)
                                                  (2)
                                                              (1)
          9    9    9    9    9    9    9    9    9    9    9    9
          0    0    0    0    0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0    1    0    1    1
          1    2    3    4    5    8    6    7    0    9    1    2
    1.4 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        |XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        |XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        |XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
D       |XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
i       |XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
u       |XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
s   0.6 +XXXXXXXXXXX    XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
t       |XXXXXXXXXXX    XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
e       |XXXXXXXXXXX    XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
r       |XXXXXXXXXXX    XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
        |XXXXXXXXXXX    XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
C       |XXXXXXXXXXX    XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
e   0.4 +XXXXXX         XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
n       |XXXXXX         XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
t       |XXXXXX         XXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXX
r       |XXXXXX         XXXXXXXXXXX    XXXXXXXXXXXXXXXXXX    XXXXXX
o       |XXXXXX              XXXXXX    XXXXXXXXXXXXXXXXXX    XXXXXX
i       |XXXXXX              XXXXXX    XXXXXXXXXXXXXXXXXX    XXXXXX
d   0.2 +XXXXXX              XXXXXX    XXXXXXXXXXXXXXXXXX    XXXXXX
s                            XXXXXX    XXXXXXXXXXXXXXXXXX    XXXXXX
                             XXXXXX    XXXXXXXXXXXXXXXX      XXXXXX
                             XXXXXX    XXXXXXXXXXXXXX
                                       XXXXXXXXXXXXXX
                                             XXXXXX
```

(1) 9007 et 9010 sont la première paire à s'unir. (Number of Clusters=11, première ligne des statistiques de PROC CLUSTER)
(2) 9006 et CL11 sont la deuxième paire à s'unir. Sachant que par CL11 on entend la fusion de 9007 avec 9010.
(3) 9005 et 9008 forment la troisième paire.

Chapitre III : LE MACRO LANGAGE

Le macro-langage est un outil qui permet de personnaliser les applications SAS à une problématique spécifique. On peut ainsi faire, avec le macro-langage, son propre langage, qui fonctionne éventuellement sur ses propres données.

Section 1 : Introduction au macro-langage

Une des utilisations les plus courantes du macro-langage est la macro-expression. Voici un exemple de macro-expression :

```
%MACRO triangle ;
Figure obtenue par l'intersection de trois droites non
parallèles
%MEND ;
```

On dit que ce texte est le développement de la macro-expression « triangle ». A l'écriture d'une macro-expression, certaines conditions doivent être satisfaites.

Le texte *texte* d'une macro-expression doit être de la forme :

```
%MACRO nom ;
texte
%MEND ;
```

où *nom* est une chaîne de caractères qui ne dépasse pas huit lettres : c'est le nom qu'on désire donner à la macro-expression.

Si nous écrivons dans un programme SAS, les trois macro-expressions suivantes :

```
%MACRO emplois ; onq oq emp tam ic %MEND;
%MACRO stages  ; sonq soq semp stam sic %MEND ;
%MACRO mesvar ; %emplois %stages %MEND ;
```

SAS va comprendre :

- par %emplois :onq oq emp tam ic,
- par %stages : sonq soq semp stam sic
- par %mesvar : %emplois suivi de %stages ; soit : onq oq emp tam ic sonq soq etc.

Enchaînons les trois lignes précédentes avec le programme suivant :

```
LIBNAME malib 'libdata' ;
PROC PRINT DATA=malib.structur ; VAR %emploi ; RUN ;
```

Pour voir dans la fenêtre « log », le compte-rendu de la manière dont SAS a développé les macro-expressions, écrire au tout début du programme :

```
OPTIONS MACROGEN SYMBOLGEN ;
```

Section 2 : Paramètres des macro-expressions

Le texte

```
%MACRO mareg ;
PROC REG ; MODEL vaest=vest ;
%MEND ;
```

rend «%mareg » équivalent au texte : « PROC REG ; MODEL vaest=vest ».

Pour faire de « vaest » un argument, SAS prévoit le symbole « & » qui signifie « le paramètre qui renvoie à ». Ainsi, dans le texte d'une macro-expression, « &vaest » signifie « le paramètre qui renvoie à vest ». Quant aux paramètres eux-mêmes, la syntaxe du macro-langage voudrait qu'ils soient annoncés entre parenthèses, séparés par des virgules juste après le nom de la macro-expression. Aussi, pour paramétrer « vaest » et « vest » dans ce qui précède, il faut écrire :

```
%MACRO mareg (vaest, vest) ;
PROC REG ; MODEL &vaest = &vest ;
%MEND ;
```

On peut donc écrire une macro-expression avec les paramètres suivants :

- *filin* pour le nom de la DATA où se trouvent les variables,
- *filout* pour le nom par lequel nous appelons la DATA où « maregre » doit stocker les résultats,
- *vaest* est la variable à estimer,

- *vest* est la liste des variables estimatives
- *ident* est l'identifiant des observations.

comme suit :

```
%MACRO maregre (filin, filout, vaest, vest, ident) ;
PROC REG DATA=&filin; MODEL &vaest=&vest;
OUTPUT OUT=&filout P=estime ;
%MEND maregre ;
```

dans la macro-expression précèdente :

- &filin : signifie : l'argument envoyé en le paramètre filin,
- &filout signifie : l'argument envoyé par le paramètre filout,
- P=estime signifie, conformément à la syntaxe de PROC REG, que la valeur estimée s'appelle « estime » dans la DATA « filout ».

Exemple d'utilisation :

```
%maregre (malib.places, sortie, cac, dj nik, mois) ;
```

Ce qui précède signifie : déclencher l'activité du macro-processeur pour l'application de la macro-expression « maregre » en mettant « malib.places » à la place du 1er argument (filin), sortie à la place du 2ème argument (filout), etc..

Exercice : Après avoir complété les points de suspension, caser le texte encadré qui suit dans la macro-expression « maregre » de manière à ce que la variable « erreur » soit, dans « filout », l'erreur commise par l'estimation.

```
DATA &filout;
SET &filout;
erreur = 100*(&... - estime)/...;
PROC PRINT DATA=&filout;var &vaest estime erreur ;
VAR &ident &vaest estime erreur ;
RUN ;
```

Section 3 : Définition d'une bibliothèque de macro-expressions

Une fois définis dans un environnement de travail un nombre important de macro expressions, il est utile de les conserver dans un répertoire extérieur qui sera référé comme lieu de dépôt des expressions ou « bibliothèque de macro expressions ». On y procède avec l'option MAUTOSOURCE. Pour la bibliothèque dont le nom UNIX est « *bibsas* », il faut écrire :

```
OPTIONS MAUTOSOURCE SASAUTOS = automac ;
FILENAME automac 'bibsas' ;
```

L'option qui précède rend SAS capable de chercher dans le filename « automac » (c'est-à-dire dans le répertoire « bibsas ») les macro-expressions non mentionnées dans votre programme.

Contrainte sur le nom physique

Par exemple, si désormais vous écrivez,

```
%f02 (toto) ;
```

SAS va chercher à exécuter la macro - expression qui se trouve dans le membre « f02.sas » du répertoire UNIX « bibsas ». Il faut donc bien retenir que :

La macro expression qui s'appelle *nom* ne peut être stockée dans une bibliothèque de macro-expressions que sous le nom de *nom*.sas

Section 4 : Macro-programmation avancée

Les macro-expressions proposées dans ce paragraphe devraient permettre à l'utilisateur de s'entraîner à rédiger des opérations nécessaires en informatique décisionnelle mais qui ne sont pas prévues dans les procédures fournies par SAS.

A] Autocorrélation de l'erreur en régression linéaire

Soit la macro - expression de l'encadré « regser ». Ce texte définit l'opération par laquelle vous appliquez la procédure PROC REG sur des variables contenues dans une DATA SAS (tableau SAS) désigné dans l'expression par « ficin ». On observe que « ficin » est le premier paramètre de l'expression et qu'il y a au total cinq paramètres, soit dans l'ordre : ficin, ficout, vaest, vest et ident.

Dans le corps de l'expression, &ficin fait référence à ficin, &ficout fait référence à ficout, etc.. En d'autres termes « &ficin » est « l'argument envoyé par ficin ». Il en va de même pour les autres paramètres (ficout, vaest, vest et ident).

```
%MACRO regser (filin,filout,vaest,vest,ident);
DATA travail; SET &filin; poids=_N_;
PROC REG DATA=travail ; MODEL &vaest=&vest;
OUTPUT OUT = &filout P=estime;

/* Calcul de l'erreur en pourcentage */

DATA &filout; SET &filout; erreur = 100*(&vaest-estime)/estime;
PROC PRINT DATA=&filout; VAR &ident &vaest estime erreur;

/* Calcul de l'erreur aux trois enregistrements précédents :
erlag1 , erlag2, et erlag3   */

DATA &filout; SET &filout;
erlag1=LAG1(erreur);
erlag2=LAG2(erreur);
erlag3=LAG3(erreur);
PROC CORR ; VAR erreur ; WITH erlag1-erlag3;
TITLE 'autocorrélation de l''erreur' ;
%MEND regser ;
```

Exercice :

- En utilisant la macro-expression « regser », estimer le Dow Jones - entre 1989 et 1992 - par le Financial Times (FT30)
- Modifier l'expression « regser » de sorte à ce qu'elle calcule l'auto-corrélation de l'erreur sur quatre temps successifs également.

B] Régression non linéaire

La macro - expression qui vous est proposée dans le fichier

bibsas/near.sas

s'utilise comme suit :

```
%near (ficin, fisor, monog, iden, td, tf) ;
```

Explication des arguments :

- « ficin » : Les données que vous voulez traiter sont dans la DATA SAS « ficin »
- « ficsor » : Les résultats de l'analyse du voisinage seront dans la DATA SAS « ficsor »
- « iden » est une variable alphanumérique d'identification des lignes,
- « monog » est la valeur de « iden » qui correspond à la ligne pour laquelle on désire calculer les plus proches voisins.
- « td » et « tf » (comme temps de début et temps de fin) : le voisinage est calculé sur les variables : COL&tb à COL&tf . Si par exemple, on a mis : td=20 et tf=30, alors, le voisinage sera calculé par référence aux variables : COL20, COL21, COL22, ..., COL30.

Contrainte sur les variables de la DATA SAS ficin :

La DATA SAS d'entrée « ficin » contient des variables de la forme : COL1, COL2, COL3, COL4, ... : ce sont les variables de référence pour l'analyse du voisinage d'une ligne particulière. Il est rappelé que les variables de la forme COL1, COL2, ... correspondent à l'option par défaut de la sortie d'un PROC TRANSPOSE.

Exercice :

- Déterminer le voisinage du CAC dans le fichier ficstag/places (ou depuis la librairie que vous avez créée)
- Appliquer la procédure PROC REG au voisinage optimal du CAC.

C] Calcul d'une tendance

Si vous disposez d'une DATA SAS « filin » dont les variables sont des séries chronologiques portant des noms de la forme : COL1,COL2, ... vous obtiendrez les paramètres de la tendance (avec les noms « pente », « reste » et « r2 ») , entre les temps i et j dans la DATA SAS « filout», en écrivant :

```
%tendance (filin, filout, i, j,pente,reste,r2) ;
```

Attention, la macro-expression « tendance » suppose que le paramètre filin comporte des variables de la forme COL1, COL2, COL3, etc. C'est la forme des variables fournies par la DATA en sortie de PROC TRANSPOSE. Aussi, si nous voulons appliquer %tendance à un fichier de la librairie « libdata », il faudra qu'au préalable, nous ayons transposé la DATA de manière à ce que les variables soient de la forme COL1, COL2, etc. alors que les observations seront, chacune, une série chronologique.

Le texte ci-dessous de la macro-expression « tendance » est disponible dans bibsas/tendance.sas.

```
%MACRO TENDANCE (FILIN,FILOUT,DEB,FIN,PENTE,RESTE,R2);
DATA &FILOUT;SET &FILIN;
      ARRAY ARY(*) COL&DEB-COL&FIN;ARRAY ARX(*) V&DEB-V&FIN;
      BASE=COL&DEB;
      DO I=1 TO DIM(ARX);ARX(I)=I;ARY(I)=ARY(I)/BASE; END;
      MEANX=MEAN(OF V&DEB-V&FIN);
      MEANY=MEAN(OF COL&DEB-COL&FIN);
      NUM=0;DEN=0;
      DO I=1 TO DIM(ARX);
        NUM=NUM + (ARX(I)-MEANX)*(ARY(I)-MEANY); *COVARIANCE;
        DEN=DEN + (ARX(I)-MEANX)*(ARX(I)-MEANX); *VARIANCE DES
TEMPS;
      END;
      &PENTE=NUM/DEN;
      &RESTE=MEANY-&PENTE*MEANX;
      ARRAY PRE(*) PRD&DEB-PRD&FIN;
      DO I=1 TO DIM(PRE);
        PRE(I)=&PENTE*ARX(I)+&RESTE;
      END;

      ESS=0;TSS=0;
      DO I=1 TO DIM(PRE);
        ESS=ESS+(ARY(I)-PRE(I))*(ARY(I)-PRE(I));*ERROR SUM OF
SQUARES;
        TSS=TSS+(ARY(I)-MEANY) *(ARY(I)-MEANY) ;*TOTAL SUM OF
SQUARES;
      END;
      IF TSS=0 THEN &R2=1000; ELSE
      &R2=1-(ESS/TSS);
      &R2=ROUND(&R2,.01);
%MEND;
```

Exercice : Appliquer l'expression à un fichier de votre choix puis utiliser cette expression pour en écrire une autre (qui l'appelle). Cette dernière devrait tester la stabilité de la variable « pente », 4ème argument de l'expression ci-avant.

D] Polysémie dans le « & »

Dans une macro expression de la forme

```
%MACRO imprimer (tableau) ;
PROC PRINT DATA=&tableau ;
RUN ;
```

le symbole « & » de « &tableau » s'entend « l'argument envoyé par tableau ». Le symbole « & » revêt cependant en SAS d'autres acceptions qui pourraient tromper l'utilisateur. Il en est ainsi quand nous écrivons :

```
%LET deb=1 ;
```

L'ordre %LET ci-dessus fait que désormais, partout dans la session, dire &deb équivaut à dire le caractère « 1 ». C'est un « & » que nous pourrions appeler direct par opposition au & argumenté de &tableau.

Annexe N°1 : Création d'un tableau de bord personnel

Des fichiers de données et d'exercices associés à ce cours sont chargés sur le réseau SUN de l'université de Paris X - Nanterre. Il faudra les copier avant de commencer les exercices.

Copie des fichiers supports du cours

Copier sous UNIX depuis la racine : **export/home/users/hathout/coursas** les répertoires et fichier suivants :

- « **fictag** » est un répertoire qui contient des données qui seront utilisées comme support des exercices. Son contenu est exposé dans l'annexe 2.
- **« bibsas** » est une bibliothèque (répertoire) de macro - expressions qui sera utilisée et complété dans la parti « macro langage »
- **« exemples** » est un répertoire qui contient des exemples. Le nom d'un membre du répertoire « exemples » renvoie au mot-clé de SAS qui y est expliqué.

Annexe N°2 : Contenu des membres du répértoire ficstag

Le répertoire « ficstag » que vous venez de copier contient des fichiers UNIX que vous serez amené à utiliser pendant le cours. Les contenus et formats d'écriture desdits fichiers sont expliqués ci-après.

ficstag/banque

Le fichier fictag/banque contient 11 observations. Les données correspondent à des chiffres de la comptabilité bancaire française, entre 1978 et 1982. Les cinq variables de chaque ligne correspondent respectivement aux années 1978, 1979, …, 1982. Chaque observation commence par une variable chaîne de caractères dont le sens est le suivant :

- cctf = credit court terme francs'
- cltf = credit lt francs'
- cmtf = Credit moyen terme francs'
- cmtd = Credit moyen terme devises'
- cltf = Crédit long terme francs'
- cltd = Credit long terme devises'
- oper = operations interbancaires'
- cctd = credit CT devises'
- ptft = Portefeuille devises'
- dev = devises'
- cltd = credit LT devises'

Format de lecture : libre (séparation des variables par des blancs). Il sera donc possible de lire ces données dans donner de format, comme suit :

```
DATA madata ;
INPUT ident an78 an79 an80 an81 an82 ;
```

ficstag/emploi

Ce fichier contient des données relatives à la formation dans l'entreprise, en France, entre 1981 et 1988. Chaque observation correspond à un agrégat « année*taille*secteur ». Les données proviennent du traitement de la déclaration fiscale 2484 par le CEREQ (Centre d'Etudes et de Recherches sur les Qualifications). Pour des raisons administratives et de secret statistique, certains secteurs ne figurent pas dans le fichier.

Variables du fichier fictag/emploi :

- année (abrégée sur 2 positions , par exemple, 81 pour 1981)
- taille des entreprises de l'agrégat (1 à 5)
- secteur d'activité économique APE 40B. Cf. annexe n°3.
- nombre de salariés hommes
- nombre de salariés femmes
- masse des salaires
- dépenses de formation.

Format de lecture : libre (séparation des variables par des blancs). Il sera donc possible de lire ces données sans préciser de format, comme suit :

```
DATA madata ;
INPUT annee taille secteur hommes femmes salaires depenses ;
```

ficstag/structure

Il s'agit de la structure des emplois et des stagiaires, en France, en 1988.

Variables du fichier ficstag/lecture :

- secteur d'activité économique APE 40B. Cf. annexe n°3
- nombre d'ouvriers qualifiés
- nombre d'ouvriers non qualifiés
- nombre d'employés
- nombre de techniciens et d'agents de maîtrise.
- nombre d'ingénieurs et cadres
- les cinq variables qui précèdent concernant le nombre de salariés qui ont effectué un stage de formation professionnelle.

Format de lecture : libre. Par exemple :

```
DATA xxx ;
INFILE définition du nom physique du fichier ;
INPUT secteur onq oq emp tam ic sonq soq semp stamsic ;
```

ficstag/places

Ce fichier contient le prix mensuel d'indicateurs internationaux.

Variables du fichier ficstag/places :

- la date sur quatre positions : 9701 signifie janvier 1997
- Indice français CAC
- Indice Dow Jones américain
- Indice britannique Financial Times FT30
- Indice canadien
- Indice CBS hollandais
- Indice belge BEL
- Indice FAZ allemand
- Indice suisse
- Indice italien
- Indice espagnol
- Indice Nikkei japonais
- Indice australien
- Indice Hang Seng de Hong Kong

Il peut être lu, par exemple, comme suit :

```
DATA xxx (xxx est un mot ne dépassant pas huit lettres);
INFILE définition de la localisation physique de fictag/places ;
INPUT (mois cac dj ftm can net bel ger swi ita spa nik syd khg) (4. 13*4.) ;
```

Sur l'ordre INPUT ci-avant, (4. 13*4.) est équivalent à (14*4.) comme à (4. 4. 4. ..... 14 fois)

ficstag/money

Ce fichier contient le prix mensuel, en francs français, de huit monnaies.

Variables du fichier ficstag/money

- la date comme pour le fichier places
- prix du dollar américain
- prix du deutch mark
- prix de la lire italienne
- prix de la livre sterling
- prix de la peseta
- prix du franc suisse
- prix du dollar canadien
- prix du yen.

On vérifie sous UNIX que les variables de ce fichier ne sont séparées par aucun espace blanc. Il est donc impossible de les lire sans format. Comme chacune de ces variables occupe 4 octets, et qu'elles sont au nombre de 9, il sera possible de les lire avec le format 9*4. .

Les fichiers numérotés f01, f02, ... etc.

Ce sont des fichiers que nous utiliserons pour la macro programmation. Voici un bref descriptif de leur contenu.

Fichier (*)	Contenu du fichier
f01	Evolution de données relatives à l'emploi, en France, par secteur d'activité économique, entre 1973 et 1990, dans les entreprises d'au moins dix salariés. .
f02	Importations et exportations européennes, en 1993.
f03	Extrait des statistiques du commerce international français, entre le 1er trimestre de 1993 et le 1er trimestre 1995.
f04	Cours mensuel moyen de treize indices internationaux entre mai 1986 et septembre 1997.
f06	Investissements japonais dans 18 pays européens , entre 1985 et 1994.
f12	Données démographiques dans 14 pays du Proche-Orient
f24	Variation du PIB dans 24 pays européens, entre 1930 et 1938. Base 100 en 1929
f32	Evolution du Produit Intérieur Brut de 11 pays européens, entre 1991 et 1996.
f35	Mots caractéristiques des discours de Bill Clinton, en 1996, en six circonstances.
f37	Indicateurs de l'emploi en Grande-Bretagne entre 1986 et 1994
f42	Psychologie américaine, en 1984.
f44	Types de recettes des protections sociales européennes, en 1994
f49	Statistiques internationales sur la production de pétrole en mai 1997.

Annexe N°3 : Codes APE 40 des secteurs d'activité économiques

Code APE	Secteur d'activité
01	Agriculture, sylviculture et pêche
02	Industrie de la viande et du lait.
03	Autres industries agricoles et alimentaires
04	Combustibles, minéraux solides, cokéfaction.
05	Pétrole, gaz naturel.
06	Electricité, gaz, eau.
07	Minerais & métaux ferreux. Première transformation de l'acier.
08	Minerais, métaux et demi-produits non ferreux.
09	Matériaux de construction, minéraux divers.
10	Industrie du verre.
11	Chimie de base. Fils et fibres artificielles & synthétiques.
12	Parachimie, industrie pharmaceutique.
13	Fonderie & travail des métaux.
14	Construction mécanique.
15A	Construction de matériel électrique & électronique professionnelle.
15B	Fabrication de biens d'équipement ménager.
16	Automobile & matériels de transport terrestre.
17	Construction navale & aéronautique, armement.
18	Industrie du textile & de l'habillement.
19	Industries du cuir et de la chaussure.
20	Bois, ameublement, industries diverses.
21	Industries du papier et du carton.
22	Imprimerie, presse, édition.
23	Caoutchouc & matières plastiques.
24	Bâtiment et génie civil & agricole.
25	Commerce de gros alimentaire.
26	Commerce de gros non alimentaire.
27	Commerce de détail alimentaire.
28	Commerce de détail non alimentaire.
29	Réparation & commerce automobile.
30	Hôtels, cafés, restaurants.
31	Transports.
32	Télécommunications et postes.
33	Services marchands rendus principalement aux entreprises.
34	Services marchands rendus principalement aux particuliers.
35	Location et crédit-bail immobiliers.
36	Assurances.
37	Organismes financiers.
38	Services non marchands.

Pour des raisons de confidentialité, les secteurs hachurés ne figurent pas dans le fichier ficstag/emploi