Méthodes multivariées appliquées à l'analyse statistique en criminologie et en sciences sociales

Partim I : Programmation SAS appliquée à l'analyse des données en criminologie et en sciences sociales

Introduction

Informations préalables.

Une brève histoire de SAS

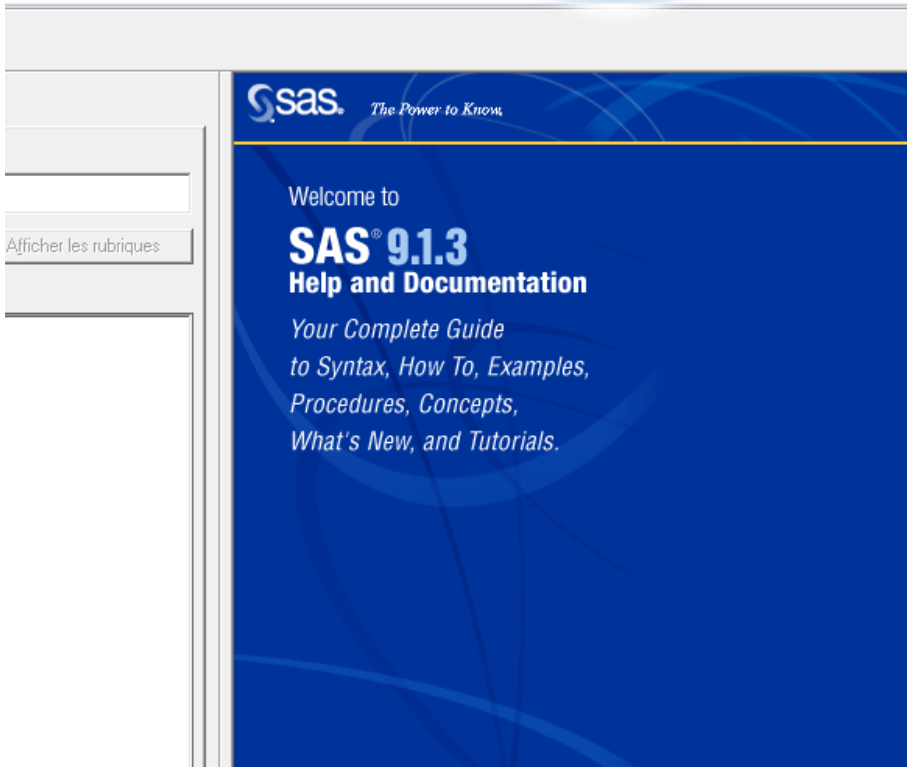
- À l'origine, SAS (*Statistical Analysis System*, Système d'Analyse Statistique) a été développé à partir de 1966 au sein de la recherche universitaire américaine pour réaliser des traitements statistiques.
- Commercialisé à partir de 1976 par « SAS Institute »
- « LE » logiciel de référence en statistiques…
  - Laboratoires pharmaceutiques,
  - Compagnies d'assurances,
  - Organismes publics, etc…
- SAS ne se vend pas mais se loue : il y a une licence annuelle. Pour les étudiants de l'ULg, c'est gratuit pendant la durée de vos études!

Où trouver SAS ?

- Sur tous les ordinateurs mis à votre disposition dans les séminaires informatiques.
- Disponible pour l'installation sur votre PC personnel.
  - → demander à l'UDI, 3ème étage de ce bâtiment.

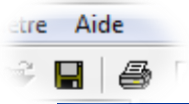
Comment apprendre à utiliser SAS ?

- En suivant ce cours;
- En utilisant l'aide en ligne;

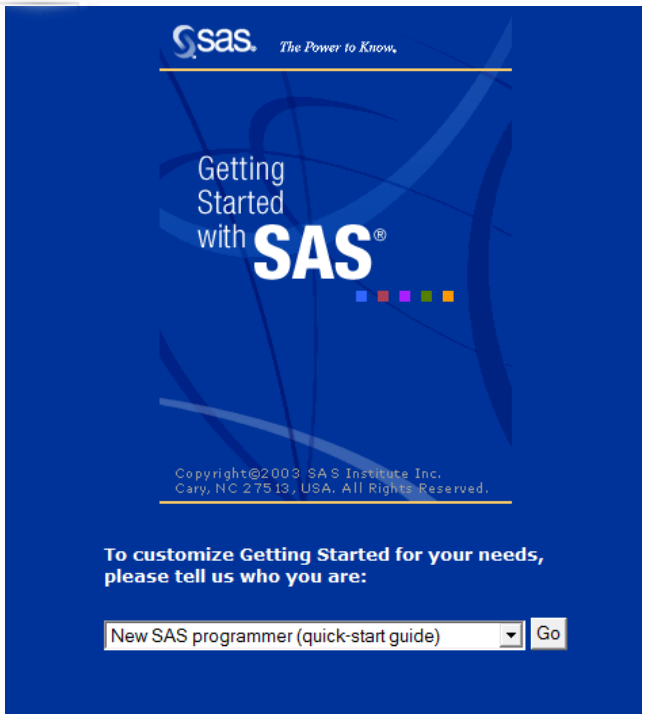


Comment apprendre à utiliser SAS ?

- En utilisant le « *Getting Started with SAS* », introduction au système SAS.

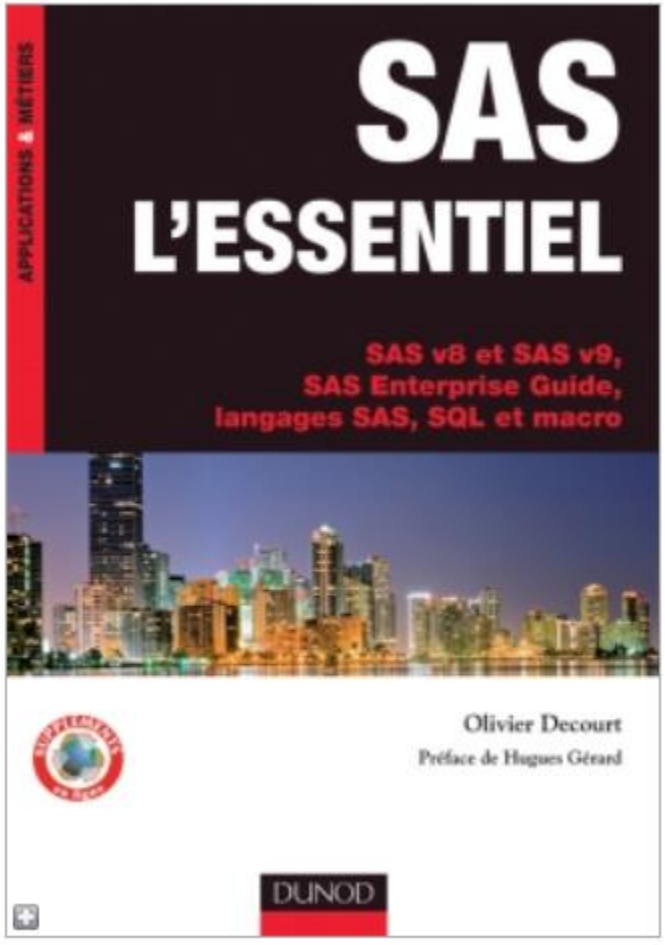


- 2 niveaux :



Comment apprendre à utiliser SAS ?

- Un livre dont je me suis inspiré pour construire ce cours (en plus des slides de Monsieur Born) :



Comment communiquerons-nous?

- En utilisant la plateforme « claroline »:

- http://campus.ishs.ulg.ac.be/
  - STAT0806-1 - Méthodes multivariées appliquées à la statistique en criminologie et en sciences sociales

    - partim I : Programmation SAS appliquée à l'analyse des données en criminologie et en sciences sociales. (20h - 3 ECTS) Cette partie du cours est consacrée à l'apprentissage du langage de programmation SAS

Chapitre 1

Prise en main de SAS

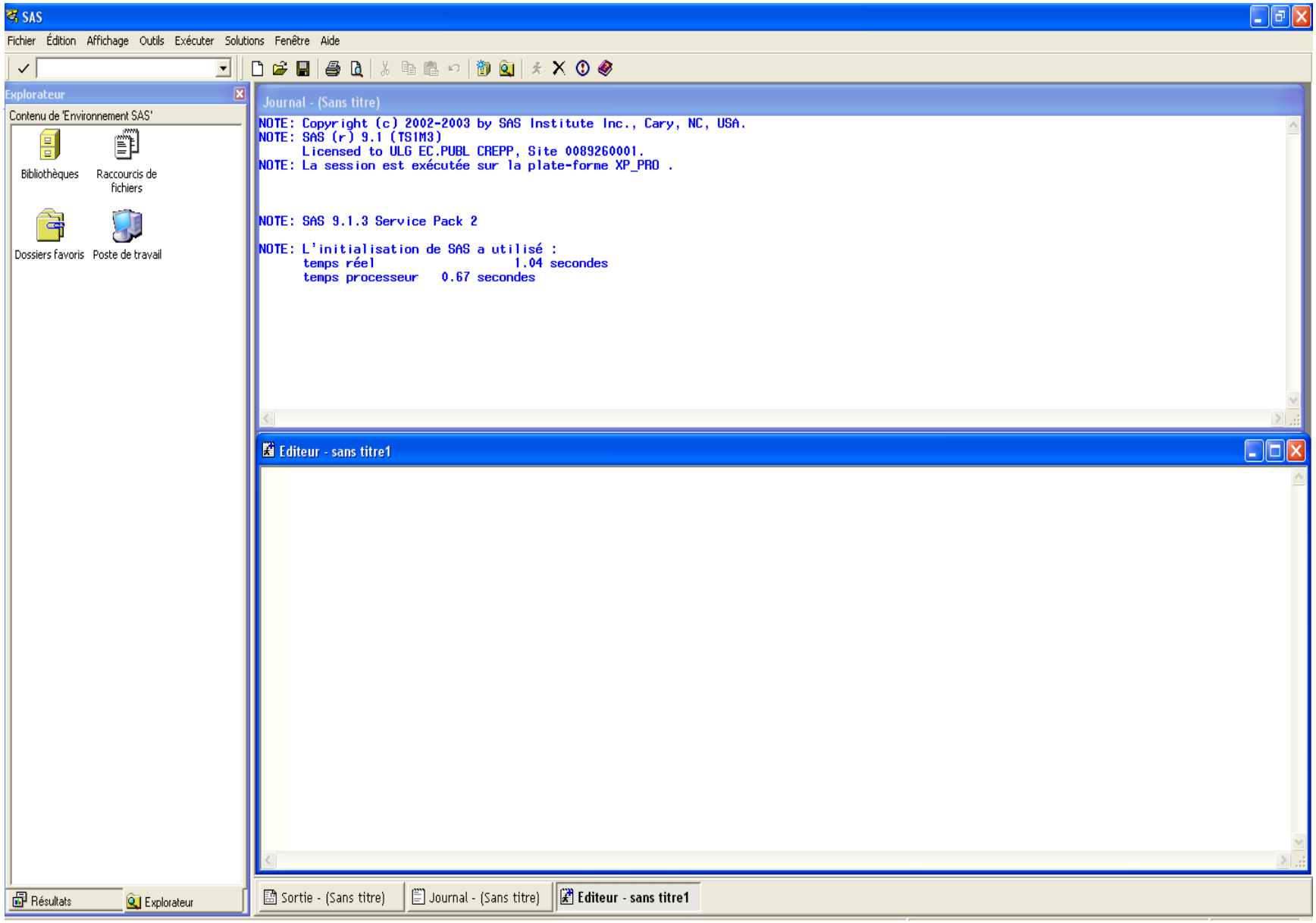
L' environnement SAS

- Démarrer SAS;
- Rappel : les fenêtres Windows (standards, options);
  - Les classiques;
  - Le « Bonhomme qui court »;
  - Le point d'exclamation;
  - Un onglet pratique : l'aide…

- Les Fenêtres SAS « Classiques », on le nomme DMS (*Display Management System*) et c'est le système gérant l'affichage. Il est composé de 5 fenêtres :

- Editeur;
- Journal;
- Sortie;
- Explorateur + Résultats (2 onglets dans une même fenêtre).

Introduction : l' environnement SAS



L' environnement SAS

- L' éditeur (*Editor*) :
  - Permet de rédiger le programme SAS.
  - L'éditeur amélioré permet de jouer sur les couleurs, les polices, de déplier ou de compacter des blocs de code.
  - Il est disponible dans le menu « Outils » - « Options » - « Préférences » Onglet « Edition »permet de sélectionner le type d'Editeur proposé par SAS; nous optons pour l'éditeur amélioré.
  - Ce choix opéré, le menu « Outils » - « Options » - « Editeur amélioré » permet d'accéder aux options spécifiques à l'Editeur, notamment les propriétés couleurs des mots du langage.
  - Tout éditeur de texte est utilisable (p.ex. le Notepad de Windows) pour rédiger un programme SAS qui sera « copié-collé » ensuite dans la fenêtre de l'Editeur.

- Le journal (*log*) : affiche les réactions de SAS à un code exécuté. On trouve 3 type de messages :
  - Les messages NOTE (en bleu) : fonctionnement normal du code;
  - Les messages WARNING (en vert) : problème à l'exécution du code mais programme non bloqué;
  - Les messages ERRORS (en rouge) : une erreur a stoppé l'exécution de ce bloc de code.

L' environnement SAS

- Les sorties (*outputs*) : affiche les résultats sous forme de texte.
- L'explorateur (*explorer*) : permet de naviguer dans les données SAS.
- Les résultats (results) :
  - Contient un sommaire des sorties;
  - Permet de naviguer entre les « outputs » et les « graphs » qui est un type de sortie particulier, affiché dans une fenêtre supplémentaire.

Programmation en SAS - Introduction

Les Mots SAS

1. **Noms** : ensembles de caractères commençant par une lettre ou l'underscore

(Max. 32767 mais la plupart sont limités à 32, voire 8).

Un nom ne peut contenir d'espace, de caractères spéciaux (sauf l'underscore); les minuscules sont équivalents aux majuscules (non sensible à la casse).

Deux types : ceux fournis par SAS (mots-clés ou réservés) et ceux créés par le programme(ur).

Un nom SAS désigne des objets qui sont traités par le programme. Un nom SAS fait référence à un emplacement de la RAM (adresse).

Programmation en SAS - Introduction

Un nom SAS peut indiquer :

- Une variable
- Un fichier
- Une procedure SAS
- Une fonction
- Un tableau
- Une étiquette d'intruction
- Une librairie
- Un format
- Une macro SAS (ce nom début alors par un & ou des %)
- …tout nom réservé par SAS

Exemples : myvar, my_var, myvar12, bb.myfile (!!!), PRINT, SqRT, _123

Si deux noms se suivent dans une instruction, la présence d'au moins un espace est obligatoire

Programmation en SAS - Introduction

2. **Littéraux** : ensembles de caractères compris entre simples ou doubles apostrophes (caractères identifiants)

Ex. 'Elisabeth' ,"a§"

3. **Nombres** : intègrent les ensembles de chiffres signés ou non, avec point décimal ou non (y compris la notation scientifique) et les dates

Ex. 23, -56.98, -1.3 E-2, '05oct09'd

4. **Caractères spéciaux** : tout caractère différent d'une lettre, d'un chiffre, du caractère de soulignement et de l'espace

Ex. +, =, /

Dans un certain nombre de cas,

- ils constitueront des mots de deux caractères : <=, **, <>, …
- voire de trois caractères : not, and,…

Programmation en SAS - Introduction

- Programme : ensemble d'instructions exécutées en séquence. Trois types d'Instructions (exécutables) existent :
  - Entrée-Sortie;
  - Traitements;
  - Structures de contrôle.

- **Documenter du code** : il s'agit d'une portion du programme qui n'est pas interprétée pas SAS.
  - Les commentaires commencent par /* ;
  - Ils se terminent par */ ;
  - Ils peuvent s'étaler sur plusieurs lignes …
  - Ils peuvent contenir du code qui ne sera donc pas exécuté.

Programmation en SAS - Introduction

Structure simplifiée d'un programme SAS : 2 blocs (« steps »)

1. Gestion des données : le DATA step ;

{Instructions de programmation} (en ce compris les structures de contrôle)

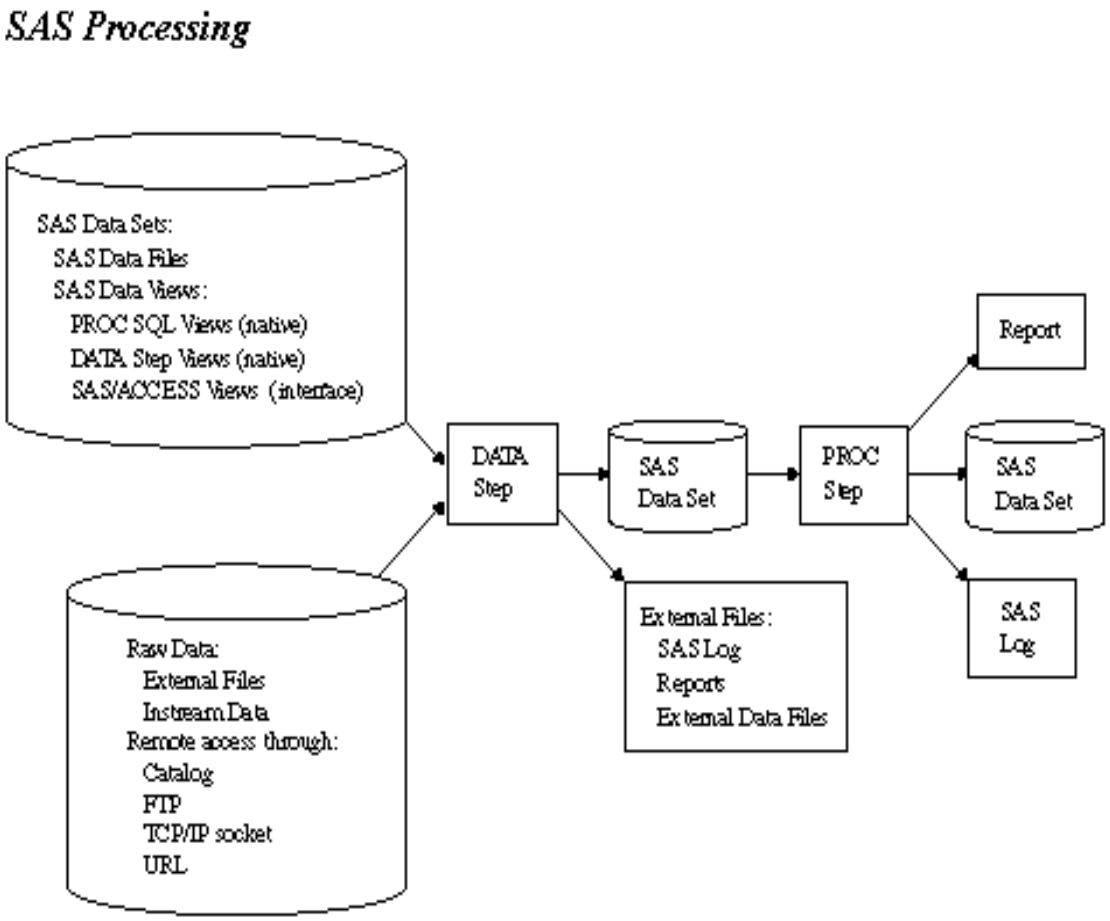
2. Traitements proprement dits : le PROCEDURE step

Une instruction de procédure commence par PROC suivi du nom du traitement à réaliser, elle renferme (plupart des cas) des:

- Options de procédures;
- Instructions de procédures (paramétrages).

Un programme peut répéter plusieurs fois cette structure mais peut aussi se limiter à une des deux étapes.

Programmation en SAS - Introduction



Source : SAS OnlineDoc, SAS Institute

Programmation en SAS - Introduction

Écrire un programme :

- L'écriture du programme est réalisée dans la fenêtre « Editeur ».
- Une instruction commence à n'importe quel endroit d'une ligne et peut se poursuivre à la ligne suivante; les espaces sont permis sauf s'ils coupent un mot.
- Une même ligne peut contenir plusieurs instructions.
- **Une instruction se termine par « ; »** .
- Si SAS peut déterminer la fin d'un mot (en général, grâce à la présence d'un caractère spécial) un espace n'est pas nécessaire mais des espaces sont conseillés pour la lisibilité.

Ex. z=10+x;

input x y z ;

drop note1 note2 ;

- Si le programme contient au moins une PROC, il se termine par l'instruction **run.**
- L'exécution d'un programme SAS est réalisée grâce au menu « Exécuter » et à l'option « Soumettre ».
- Il est possible de n'exécuter qu'une partie du code en sélectionnant les lignes correspondantes.

Chapitre 2

Le « DATA step »

Programmation en SAS : le DATA step

- Ensemble des instructions qui définissent le fichier de données en format SAS (« sas data set »).

  Un fichier de données SAS se compose :

  - des données proprement dites (tableau statistique : variables en colonnes et observations en lignes)

  - d'un dictionnaire des données qui contient toutes les informations qui caractérisent le fichier.

- Les données sont imprimées par la PROC PRINT.
- Le dictionnaire s'affiche par la PROC CONTENTS.

- Le DATA step peut contenir les encodages (les données du tableau statistique).
- Les données sont le plus souvent disponibles sous forme de fichiers préexistants (de format texte, excel, autres…). Dans ce cas, SAS fournit les outils adéquats d'importation

Programmation en SAS : le DATA step

```
/* Structure d'un programme rédigé en SAS*/
/*La partie data : gestion des données*/
data;
input idnumber $ week1 week16;
weightLoss2 = week1-week16;
datalines;
2477 195 163
2431 220 198
2456 173 155
2412 135 116
;
proc contents; /*essayons aussi proc print;*/
run;
```

- Si on a oublié à quoi servent « data », « input »,…
  → On utilisera l'aide…

Programmation en SAS : le DATA step

- Les données sont imprimées par la PROC PRINT.
- Le dictionnaire s'affiche par la PROC CONTENTS.

Programmation en SAS : le DATA step

Impression du fichier
(tableau statistique)

Le Système SAS 10:15 Tuesday, August 28, 2007 2

Obs	idnumber	week1	week16	weight Loss2
1	2477	195	163	32
2	2431	220	198	22
3	2456	173	155	18
4	2412	135	116	19

Rédiger et exécuter ce programme

Programmation en SAS : le DATA step

Impression du dictionnaire

Le Système SAS 10:15 Tuesday, August 28, 2007 1

La procédure CONTENTS

Nom de la table SAS	WORK.DATA1	Observations	4
Type d'entrée	DATA	Variables	4
Moteur	V9	Index	0
Créé(e)	mardi 28 août 2007 10 h 52	Longueur d'observation	32
Dernière modification	mardi 28 août 2007 10 h 52	Observations supprimées	0
Protection		Compressé(e)	NON
Type de table		Trié(e)	NON
Libellé			
Représentation données	WINDOWS_32		
Codage	wlatin1 Western (Windows)		

Informations propres à l'hôte/au moteur

Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	126
Obs in First Data Page	4
Number of Data Set Repairs	0
Nom de fichier	C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS Temporary Files_TD3564\data1.sas7bdat
Date de création de la version	9.0101M3
Date de création de l'hôte	XP_PRO

Liste alphabétique des variables et attributs

#	Variable	Type	Long.
1	idnumber	Alph	8
2	week1	Num	8
3	week16	Num	8
4	weightLoss2	Num	8

Programmation en SAS : le DATA step

- Le DATA step peut contenir les encodages (les données du tableau statistique).

MAIS…

- Les données sont le plus souvent disponibles sous forme de fichiers préexistants (de format texte, excel, autres…). Dans ce cas, SAS fournit les outils adéquats d'importation

Programmation en SAS : le DATA step

Création d'un fichier de données

INPUT v1 v2 v3…;

est l'instruction qui permet de créer les colonnes du tableau statistique (les champs du fichier) et d'y enregistrer les valeurs des lignes (les enregistrements du fichier) .

INPUT est un nom réservé (SAS « keyword »)

Les colonnes sont les variables, les lignes constituent les observations.

Noms de variables : 32 caractères max. le premier est alphabétique ou le souligné, pas d'espaces, pas de caractères spéciaux, majuscules=minuscules .

Types : Nous allons encoder des prénoms et des poids → variable « numérique » ou* « caractère » (présence du $).

Programmation en SAS : le DATA step

1er cas possible : les données sont encodées dans le DATA step (instruction **datalines** ou **cards**)

(cf. diagramme supra « Raw data – Instream Data ») → données brutes.

```
data ;
   input name$ week1 week16;
    datalines;      /* ou cards */
Julie 195  163
Audrey 220  198
France 173  155
Judith 135  116
;
run;
```

- Nous pouvons y ajouter les procédure « proc print » et « proc contents ».

Programmation en SAS : le DATA step

Règles:

- La présence des encodages dans le programme est déclarée par DATALINES ;
- **Un point-virgule final** signale la fin des données (!)
- Assignation des valeurs dans l'ordre de la liste
- Respect du type

Nombre d'observations = nombre de répétitions du DATA step

- Dans les lignes de données, un espace (ou plus) signale la fin d'une valeur (délimiteur de champs)
- Règles pour les valeurs manquantes :
  - Par défaut, la présence du « . » (point) dans les encodages
  - Dans le fichier de données, une valeur manquante est représentée par le « . »
  - Les valeurs manquantes ne participent pas aux calculs statistiques.

Programmation en SAS : le DATA step

2ème cas possible : les données sont encodées dans un fichier existant

(cf. diagramme supra « Raw data – External Files »)

instruction **infile** 'nom de fichier';

Soit un fichier de type texte : monfichier.txt (enregistré sur mon ordi à un endroit particulier)

```
data ;
infile 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\Data\Monfichier.txt';
input name$ Week1 Week16;
proc print ;
run;
```

- L'instruction INFILE est obligatoire, elle indique le nom **complet** (entre apostrophes) du fichier qui renferme les encodages.(nom complet = endroit physique, chemin des dossiers, nom du fichier)
- Pour connaitre le nom complet et l'adresse complète d'une fichier : click droit → propriété.

Exercice : encoder un fichier texte qui contiendra les données de l'exemple ci-dessus et le lire en SAS

Programmation en SAS : le DATA step

L'instruction **INPUT** utilise l'espace comme séparateur des zones de données. Un espace signifie la fin des données pour une variable et le passage à la zone de données de la variable suivante; cependant, il arrive très souvent que des données concernant une variable de type caractère contiennent des espaces mais également que la présence d'espaces séparateurs ne soit pas garantie.

Ex.

```
data;
      input ministre $ budget;
      datalines;
      Van de Lanotte  1000
       de Villepin  2500
;
```

La présence des espaces dans les données pour la première variable (de type caractère) va provoquer une erreur : la donnée « Van » sera affectée à la variable ministre, puis l'espace provoquera le passage à la lecture de la variable suivante c.à.d budget (de type numérique). Les caractères « de » seront alors affectés à celle-ci, ce qui provoquera une erreur de type, la variable étant de type numérique.

Programmation en SAS : le DATA step

Dans ce dernier cas, on utilise dans ce cas une version différente de l'instruction INPUT (« column input ») :

INPUT v1 ($) *colonne de début – colonne de fin v2 ($) colonne de début – colonne de fin …;*

colonne de début donne la position du premier caractère de la variable et *colonne de fin* la position du dernier dans la ligne d'encodage

Ainsi dans l'exemple précédent, si on a encodé les données pour la première variable dans les colonnes 1 à 15 et celles de la deuxième dans les colonnes 22 à 25, on écrira le programme:

```
data;
      input ministre $ 1-20 budget 22-25;
      datalines;
Van de Lanotte       1000
De Villepin          2500
;
```

Programmation en SAS : le DATA step

Exemple:

Sur le site de 'European Social Survey', on peut trouver les données exhaustives des enquêtes réalisées depuis 2002.

Ces données sont téléchargeables pour des traitements locaux, notamment sous format « texte ».

http://www.europeansocialsurvey.org/

On a sélectionné les données belges relatives à 5 questions:

1. TV watching, total time on average weekday (0 No time at all 1Less than 0,5 hour 2 0,5 hour to 1 hour 3 More than 1 hour, up to 1,5 hours 4 More than 1,5 hours, up to 2 hours 5 More than 2 hours, up to 2,5 hours 6 More than 2,5 hours, up to 3 hours 7 More than 3 hours)
2. How interested in politics (1 Very interested 2 Quite interested 3 Hardly interested 4 Not at all interested)
3. How happy are you (0 Extremely unhappy 1 – 10 Extremely happy)
4. Gender (1 Male 2 Female)
5. Year of birth

Le fichier s'appelle ess 2006 subset.txt et se trouve sur le Bureau de Windows; voici les premières lignes (observations):

```
73 921939
74 821935
72 811978
53 911961
32 821954
32 311980
43 921978
```

En colonne 2 : les réponses à la question 1

En colonne 3 : les réponses à la question 2

En colonnes 4 et 5 : les réponses à la question 3

En colonne 6: les réponses à la question 4

En colonnes 7 à 10 : les réponses à la question 5

Rédiger le programme en SAS qui permet de lire ces données et créer le SAS data set

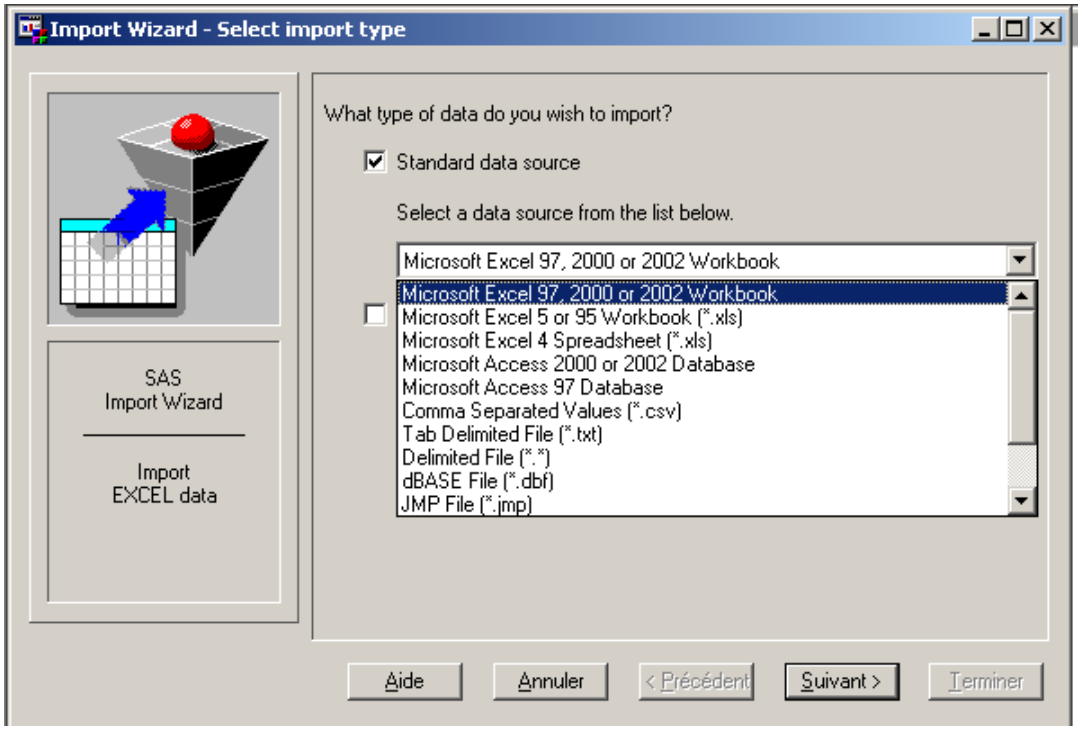
```
/* Lire les données de l'enquête ESS 2006 */
/* NB : Le nom du fichier est ess_2006_subset.txt et se trouve
dans un dossier
intitulé SAS partim I, il se trouve dans mes documents
Il faut adapter ce littéral en fonction du PC */
data;
infile 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et
2\mes cours\Partim_I_-_Le_langage_SAS\Data\ess_2006_subset.txt';
input TV 2-2
      Politics 3-3
          Happy 4-5
          gender 6-6
          year_of_birth 7-10;

          run;
```

Programmation en SAS : le DATA step

3ème cas possible : il s'agit d'importer des données d'un fichier existant

Le menu 'Fichier' →'Importer données…' permet de préciser le format du fichier dont les données doivent être importées pour la création du fichier en format SAS:



(La plupart des formats utilisés en Windows sont accessibles)

Programmation en SAS : le DATA step

Le fichier villes.xls contient les données de criminalité pour 20 communes wallonnes, ce fichier est sur le http://campus.ishs.ulg.ac.be.

Créer le SAS dataset.

Tests (1) : Majuscules

```
data;
    input IDNUMBER $ WEEk1  week1;
    datalines;
2477 195  163
2431 220  198
2456 173  155
2412 135  116
;
run;
```

Tests (2) : Lignes vides

```
data;
input idnumber $ week1 week16;


   datalines;
2477 195  163
2431 220  198
2456 173  155
2412 135  116
;
run;
```

Tests (2) : Lignes vides !!! dans les données

```
data;
input idnumber $ week1 week16;


   datalines;
2477 195  163

2431 220  198

2456 173  155
2412 135  116
;
run;
```

Tests (3) Instruction sur plusieurs lignes

```
data;
input
idnumber $
datalines;
2477 195  163
2431 220  198
2456 173  155
2412 135  116
;

run;

/* Il manque un truc…*/
```

Tests (4) Plusieurs instructions par ligne

```
data;
input idnumber $ week1 week16; datalines;
2477 195  163
2431 220  198
2456 173  155
2412 135  116
;
run;
```

Tests (5) Espaces

```
data;
input idnumber $ week1
week16;
   datalines;
2477   195                163
2431   220  198
2456   173  155
2412  135  116
;
run;
```

Tests (6) Espaces

```
data;
input id                    number $ week1
week16;
 datalines;
2477   195    163
2431    220  198
2456    173   155
2412     135  116
;

run;
```

Chapitre 3

Gestion des fichiers de données | Les bibliothèques

Les bibliothèques, introduction…

- Les données accessibles directement par SAS sont rangées dans des bibliothèques appelée très souvent « LIBRAIRIES ». La plupart du temps les librairies correspondent à un emplacement physique (sous Windows, un répertoire).

- MAIS, il existe une librairie temporaire : « WORK ». Elle est personnelle et temporaire → unique pour chaque session.
  - **Son contenu est détruit à la fermeture de la session SAS.**

Fichiers temporaires

- Tous les fichiers de données en format SAS créés jusqu'à présent par l'instruction DATA ont une durée de vie limitée à la session SAS (temps qui sépare le lancement du système SAS de la fermeture de l'application);
  - → Ce sont des fichiers temporaires.

- Ils portent un nom composé de 2 mots séparés par le point;
  - le premier est WORK et le deuxième est, par défaut, DATAn, où n est un entier qui commence à 1 et s 'incrément de 1 à chaque DATA step.

Gestion des fichiers de données en SAS

Ex. Dans le cadre d'une même session SAS:

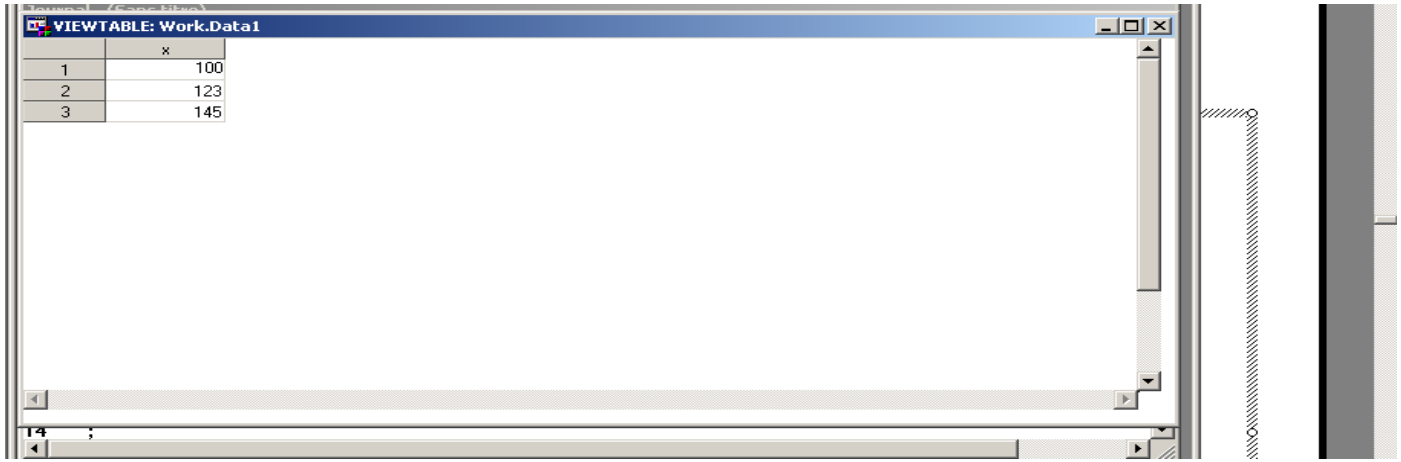
```
data;
Input x;
 datalines;
100
123
145
;
/* --> Crée le fichier work.data1 */
data;
input  name $ cours1;
datalines;
Julie 13
Tom 12
Audrey 14
;
/* --> Crée le fichier work.data2 */
```

(Voir le fichier Journal)

Gestion des fichiers de données en SAS

- WORK est le nom d'une librairie. Ce dossier est temporaire, il est supprimé dès qu'on quitte le programme SAS).
  (voir fenêtre 'Explorateur' -> 'Bibliothèque'->'Work')

- Un double clic sur le fichier (ci-dessous Data1) ouvre une fenêtre d'édition du fichier;

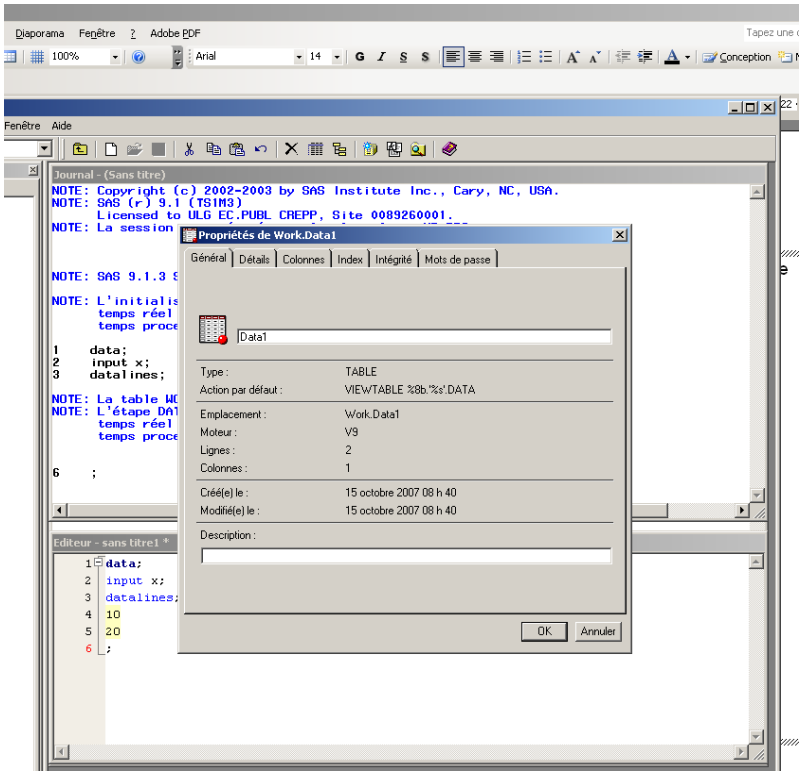


- « Viewtable » possède sa propre barre de menus qui permet de modifier le fichier sans passer par un programme.
  - Mode **édition** VS Mode **visualisation**

Gestion des fichiers de données en SAS

- WORK est le nom d'une librairie. Ce dossier est temporaire, il est supprimé dès qu'on quitte le programme SAS).
(voir fenêtre 'Explorateur' -> 'Bibliothèque'->'Work')

- Les propriétés du fichier (clic droit puis 'Propriétés') créé permettent d'afficher toutes les caractéristiques (SAS et Windows) du fichier de données.



Gestion des fichiers de données en SAS

- La fenêtre « propriété » montre que les variables (voir onglet « colonnes ») ont de nombreux attributs : nom, type, label, longueur, format…

1. Nom et label.
   Les colonnes d'une table SAS correspondent aux variables. Elles doivent être très clairement identifiées pour se rattacher facilement à l'information qu'elles contiennent.

   - Elles portent donc un NOM, celui-ci doit être unique à l'intérieur de la table. Il doit faire 32 caractères au maximum. Il doit se composer uniquement de lettres (sans accent) et éventuellement de chiffres et de l'underscore.
     Le nom d'une variable ne peut pas commencer par un chiffre.
   - Les variables peuvent aussi être agrémentées d'un label. Ce texte entièrement libre (accents, tirets et espaces acceptés) de 256 caractères maximum peut contenir une description plus précise de la variable.

2. Type.
   Les variables ont aussi un type : numérique ou caractère.
   Si elle est **numérique**, la variable ne peut contenir que des nombres → toutes les opérations mathématiques et statistiques sont autorisées sur cette colonne.
   Si elle est de type **caractère [$]**, la variable peut contenir tout type de valeur, texte ou non, mais les opérations statistiques ne sont pas autorisées.

Gestion des fichiers de données en SAS

- La fenêtre « propriété » montre que les variables (voir onglet « colonnes ») ont de nombreux attributs : nom, type, label, longueur, format…

3. Format et longueur
   - Le format permet d'afficher les valeurs selon une forme spécifique, par exemple, pour les dates.
   - La longueur est le nombre d'octets mobilisés pour le stockage de chaque valeur de la variable.

   SAS définit des longueurs par défaut mais il est conseillé d'optimiser ce point pour des tables très volumineuses

Gestion des fichiers de données en SAS

- Une instruction PROC traite par défaut le dernier fichier créé:

```
/* premier */
data;
Input x;
 datalines;
100
123
145
;
/* second */
data;
input  name $ cours1;
datalines;
Julie 13
Tom 12
Audrey 14
;

Proc print;
run;
```

→ affiche le contenu du dernier data;

Gestion des fichiers de données en SAS

- Le traitement par défaut du dernier fichier créé peut être modifié par l'instruction: options _last_= nom de fichier;
  où _last_ est une variable du système SAS qui enregistre le nom du dernier fichier de données.
  (nb: tous les mots réservés en SAS qui commencent et se terminent par _ sont des variables système)
- L'instruction SAS options est appelée globale car on la retrouve aussi bien dans la partie DATA que dans la partie PROC. Elle est active jusqu'à l'apparition d'une autre instruction options OU la création d'un nouveau SAS dataset.

```
data;
Input x;
 datalines;
100
123
145
;
data;
input  name $ cours1;
datalines;
Julie 13
Tom 12
Audrey 14
;
options _last_= work.data1;    /* on peut aussi écrire : options _last_= data1 */
Proc print;
run;
```

Gestion des fichiers de données en SAS

- Dans la partie DATA du programme, l'accès à un fichier se fait par l'instruction d'appel de fichier :
  SET nom de fichier;
  Le résultat est le transfert des données de ce fichier:

```
data;
set data1 ;     /* ou : set  work.data1 */
```

→ crée un nouveau fichier à partir des données de data1 qui peuvent ensuite être modifiées. Le fichier temporaire créé par l'instruction data ci-dessus n'a pas pour nom 'work.data1' (cf. noms des fichiers de données supra)

- On peut spécifier dans l'instruction DATA un nom de fichier :

```
DATA monfichier;
```

→ A pour effet d'attribuer au fichier que l'on crée le nom monfichier; ce fichier conserve néanmoins le statut de temporaire (son nom est work.monfichier).
Le nom choisi doit évidemment respecter les règles de SAS (syntaxe et longueur max. de 32 caractères).
L'accès se réalise par SET monfichier; (ou set work.monfichier;)

Gestion des fichiers de données en SAS

Exemple: On a importé le fichier villes.xls et on l'a appelé villes, quel est le résultat de :

```
data;
set villes;
/*(cf. la fenêtre Journal)*/

/*Ensuite, que se passe-t-il avec l'instruction suivante ? */
proc print;
run;
```

Enfin, si nous entrons l'instruction suivante?

```
data;
set monfichier;
proc print;
run;
```

Gestion des fichiers de données en SAS

Très souvent, l'utilisateur de SAS a envie de pouvoir utiliser ses bases de données lors de plusieurs session SAS. Il doit alors créer de **Fichiers permanents.**

- **Création**

L' instruction DATA complétée par un nom de fichier placé entre (simples) apostrophes enregistre le fichier de données.

data 'fichier';

input x y ;

datalines;

10 20

22 55

;

Crée le fichier de données 'fichier' dans un dossier par défaut parfois difficile à localiser.

L'encodage du nom complet du fichier reste préférable; pour créer le fichier sur le Bureau de Windows, on vérifiera les propriétés de 'Bureau ' et on spécifiera, selon le poste utilisé par

Pour mon exemple:

```
data 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\fichier8oct';
input x y ;
datalines;
10 20
22 55
;
```

Gestion des fichiers de données en SAS

Accès

Dès que le fichier a le caractère permanent, il est accessible dans le DATA step par l'instruction:

SET nom du fichier ; (présence des apostrophes obligatoires)

Ex.

data;

```
set 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et
2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\fichier8oct';
```

Exemple: créer le fichier des données ESS (reprendre le programme de création)

- Le fichier s'appelle ESS2006
- Il se trouve dans le dossier appelé DATA sur le Bureau (ou à un autre endroit précis!)
- Il est permanent

Gestion des fichiers de données en SAS

```
Data 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1
et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ESS2006';

infile 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim
1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\Data\ess_2006_subset.txt';

input
        TV 2-2
        Politics 3-3
        Happy 4-5
        gender 6-6
        year_of_birth 7-10;

          run;
```

- (Nb:rôle des littéraux)

Gestion des fichiers de données en SAS

- **les librairies**

Dans l'instruction data, le nom du fichier est composé de deux mots séparés par un point :

- Le premier est un nom de librairie qui fait référence à un dossier (cf. supra) créé antérieurement dans l'environnement Windows,
- le deuxième mot (nom SAS) est le nom du fichier proprement dit.

Cette référence librairie SAS (dossier WiINDOWS) se fait dans l'instruction :

libname *nom de librairie nom de dossier;*

- Cette instruction **DOIT** précéder l'instruction data dans le programme.

- *nom de librairie* est un nom SAS limité à 8 caractères
- *Nom de dossier* est un nom Windows entouré d'apostrophes (littéral SAS)

Gestion des fichiers de données en SAS

Dans l'exemple, ci-dessus, **on commence par créer le dossier sas** sur le Bureau de Windows, puis on y enregistre les données dans l'environnement SAS:

```
libname malib 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest';
data malib.fiche;
input x y ;
datalines;
10 20
22 55
;
```

- malib est le nom de la librairie qui contient le fichier 'fiche';
- Une seule instruction **libname** suffit pour tout le programme;
- Tout fichier dont le nom commence par le nom de la librairie est permanent et stocké dans le dossier référencé par la librairie.

Gestion des fichiers de données en SAS

La combinaison des instructions libname et set permet l'accès à un fichier de données créé antérieurement.

Exercice:

Créer le fichier des étudiants à ce cours:
-Nom
-Prénom
-Matricule
-Age

Les données sont encodées dans le programme et le fichier s'appelle exercice.student
Une fois créé, on quitte l'application SAS.

…solution

```
libname exercice 'C:\Users\SEBA Fontaine\Documents\dropbox\mes
cours\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\pourtest';
data exercice.student;
input nom$ prenom$ matricule$ age;
datalines;
Dupont Jean s201001 20
Camus Paul s200988 23
;

On peut sortir de SAS et essayer au retour :


libname exercice 'C:\Users\SEBA Fontaine\Documents\dropbox\mes
cours\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\pourtest';
data exercice.student2;
set exercice.student;
proc print;
run ;
;
```

- ***Modification des lignes et des colonnes (mise à jour)***

- **instructions : update fichier1 fichier2 ; by var;**

Où

- **fichier1 est le fichier à modifier**
- **fichier2 contient les valeurs pour la mise à jour (fichier des transactions)**
- **var est la variable d'identification des lignes dans les deux fichiers (obligatoire!)**

- **C'est le deuxième fichier qui a la main (s'il y a conflit).**

Exemple de mise à jour

```
libname student 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest';
data student.one;
input mat $ prenom $ cours1 cours2;
datalines;
s001 . 12 13      /* A la création de ce fichier, le nom du premier étudiant n'est pas connu */
s002 Julie 12 .  /* de même que la valeur de cours2 pour cet étudiant  */
s003 Tom 11 10
;
data student.two;
input mat $ prenom $ cours1 cours2;
datalines;
s001 Laura . .     /* On crée un fichier qui contient les mises à jour pour les deux étudiants */
s002 . . 15
;
data student.one;
update student.one student.two; by mat;    /* Mise à jour du premier fichier créé */
run;
```

Gestion des fichiers de données en SAS

- **Fusion de Fichiers**

L'opération de fusion consiste à créer un fichier par addition de fichiers existants.
On peut considérer deux cas possibles:

I. Les fichiers fusionnés renferment des observations différentes et des variables identiques.
On ajoute des lignes dans une base de données.

Par exemple …

```
data;
input x y ;
datalines;
100 250
150 300
;

data;
input x y;
datalines;
200 300
250 400
300 450
;
```

Gestion des fichiers de données en SAS

Le premier data step crée un fichier de données work.data1 et le deuxième un fichier work.data2;
on désire créer un seul fichier qui contient les 5 observations pour les variables x et y

L'instruction **set** fichier1 fichier2….;
fusionne les fichiers dans le sens des lignes

```
 data total;
set work.data1 work.data2;
run;
```

- Le fichier « total » créé est la concaténation des deux fichiers en amont.
- Exécuter le programme en SAS et vérifier le log

Gestion des fichiers de données en SAS

2.Les fichiers fusionnés renferment des variables différentes pour les même observations. On ajoute des colonnes dans une base de données.

Ex.

```
data;
input v1 v2 v3 ;
datalines;
1000  250 10
1500  300 20
;

data;
input v4 v5;
datalines;
1200 2300
1250 3400
;
run;
```

Gestion des fichiers de données en SAS

L'instruction **merge** fichier1 fichier2….;
fusionne les fichiers dans le sens des colonnes (des variables)

Le programme de fusion des fichiers s'écrira:

```
data;
merge work.data3 work.data4;  /* ou merge data1 data2  ; */
Run;
```

Ce data step crée un fichier de 2 observations et 5 variables dont le nom est work.data3

Exercice

- Créer 2 fichiers :
  - le premier contient le nom de la commune et la population

communes	popul
Seraing	67000
Liège	200000

  - le second, la superficie et le nombre de policiers

superf	nbpol
2000	500
1000	1200

- Créer un troisième fichier permanent qui contient toutes les variables ;
- On utilise une librairie appelée ‘ communes’

Gestion des fichiers de données en SAS

```
libname communes 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\';
data communes.one;
input communes $ popul ;
datalines;
Seraing 67000
Liège 200000
;
data communes.two;
input superf nbpol;
datalines;
2000 500
1000 1200
;
data communes.final;
merge communes.one communes.two;
proc print; /* va nous donner une sortie; */
run;
```

Gestion des fichiers de données en SAS

merge fichier1 fichier2….; **(suite)**

Soit deux fichiers:

(la première colonne identifie les lignes communes)

Le deuxième fichier ne contient pas les mêmes observations.

La technique précédente conduit à un résultat non escompté.

0001	Seraing	61000
0002	Liège	186000
0003	Neupré	1500

La solution:

merge fichier1 fichier2….;

BY var**;**

! Les identifiants des lignes communes doivent être rangés dans le même ordre

(Le programme SAS se trouve dans la dia suivante)

0000	1000
0001	2000
0002	1200
0003	500
0004	120

Gestion des fichiers de données en SAS

```
data one;
input id $ nom $ population;
datalines;
0001 Seraing 61000
0002 Liège 186000
0003 Neupré 1500
;
data two;
input id $ nbpoliciers;
datalines;
0000 1000
0001 500
0002 1200
0003 500
0004 120
;
data all;
merge one two; by id;
proc print ; run;
```

Gestion des fichiers de données en SAS : Synthèse à propos de la fusion de fichiers

- Pour **ajouter des lignes** (observations) : set fichier1 fichier2 …;

Le fichier créé contient les lignes de fichier1, fichier2, … dans l'ordre où ceux-ci apparaissent dans l'instruction de fusion.

Si les colonnes (variables) diffèrent entre les fichiers fusionnés, il y a création de valeurs manquantes

- Pour **ajouter des variables** (colonnes), deux cas sont possibles:
  - les fichiers à fusionner contiennent des lignes qui font référence aux mêmes observations : merge fichier1 fichier2 … ;
  - les fichiers à fusionner contiennent des lignes qui se réfèrent à des observations différentes : merge fichier1 fichier2 … ; by var;

où var est la colonne qui identifie les lignes

les fichiers doivent avoir été triés sur cette variable d'identification

Gestion des fichiers de données en SAS

```
data one;
input id $ nom $ population;
datalines;
0001 Seraing 61000
0002 Liège 186000
0003 Neupré 1500
;
data two;
input id $ nbpoliciers;
datalines;
0000 1000
0001 500
0002 1200
0003 500
0004 120
;
data all;
merge one two; by id;
proc print ; run;
```

Gestion des fichiers de données en SAS

Instructions additionnelles pour la mise à jour de fichiers

delete ; forme équivalente : **remove;**

exclut l'observation en cours du fichier

keep *v1 v2 v3 …;*

enregistre les colonnes (variables) dont les noms figurent dans la liste

drop *v1 v2 v3 …;*

supprime les colonnes (variables) dont les noms figurent dans la liste

rename *v1=v1new v2=v2new …..;*

modifie le nom des colonnes (variables)

label *v1 = 'descriptif var1' v2 = 'descriptif var2' …. ;*

associe des étiquettes aux variables (max. 256 caractères)

Exercice:Dans l'exemple ci-dessus, la variable nbpoliciers s'appelle désormais police et on lui donne une étiquette 'Effectifs de police'.

Chapitre 4

Expressions et assignations

Expressions et assignations

- **Création d'un variable par assignation**

Tous les langages de programmation disposent de la possibilité de créer des variables, donc de nouvelles entités stockées en mémoire centrale.

En SAS, l'instruction de création de variable répond à la syntaxe :

v = expression ;

v est le nom de la variable que l'on crée (max. 32 caractères + règles de syntaxe)

Une variable de programme = un emplacement en mémoire Ram

ex. vtotal=v1+v2 ;

vmean=(variable1+variable2)/2;

nom = 'Dupond et Dupont' ;

Expressions et assignations

- **Les expressions**

En informatique (comme en algèbre), une expression est une combinaison d'opérandes et d'opérateurs dont l'évaluation aboutit à une valeur (le résultat).

Puisque les instructions comprises dans le DATA step sont exécutées autant de fois qu'il y a d'observations, l'évaluation de l'expression produit un résultat pour toutes les lignes du tableau.

ex. vtotal=v1+v2 ;

vmean=(variable1+variable2)/2;

nom = 'Dupond et Dupont' ;

Contrairement à l'algèbre, les opérandes ne sont pas nécessairement numériques, les opérateurs ne se limitent pas aux opérateurs arithmétiques habituels et, en conséquence, le résultat peut être de plusieurs types: numérique, caractère, logique.

Opérandes : constantes, variables, fonctions internes

Opérateurs : arithmétiques, relationnels, logiques

Expressions et assignations

- **OPERANDES :** Entités sur lesquelles les opérateurs s'appliquent

  - *Constantes: (*valeurs fixes pendant le déroulement du programme)

    - Numériques : entiers ou décimaux, signés ou non selon la notation usuelle ou scientifique (-4, 1.23, 100, 1e2, -0.1e-2);
    - Caractères : suite de caractères entre apostrophes (max. 32767);
    - Dates.

  - *Variables:* (emplacements en mémoire dont le contenu change au cours de l'exécution du programme)

    - Numériques;
    - Caractères;
    - Dates.

  - Fonctions internes: Petits programmes appelables par leurs noms (mots-clés SAS) qui réalisent des traitements spécifiques, ils enrichissent les possibilités de traitement sans imposer des travaux de programmation.
    - Analogie : touches fonctions d'une calculette

Expressions et assignations

Principales catégories de fonctions

- Traitements des Tableaux
- Traitements de chaînes de caractères
- Traitement des dates
- Statistiques descriptives
- Calculs financiers
- Mathématiques, probabilités et trigonométrie
- Génération de nombres au hasard
- Gestion des fichiers
- Arrondis

Expressions et assignations

- Appel d' une fonction dans un programme :

nom de fonction (arg1, arg2, arg3,…)

  - où arg1, arg2,arg3 sont les arguments (constantes, variables et expressions)
  - pour les fonctions sans argument, les parenthèses sont obligatoires

<u>Exemples de traitement des chaînes de caractères :</u>

- length (arg) où arg est de type caractère : renvoie la longueur de la chaîne

length('Ceci est un exemple') vaut 19

- lowcase (arg) où arg est de type caractère : convertit arg en minuscules

lowcase('MAJUSCULES') vaut 'majuscules'

- substr (arg1,arg2,arg3) : arg1 est une chaîne, arg2 et arg3 sont des valeurs entières

la fonction renvoie les arg2 caractères de arg1 en commençant à arg3

substr ('apples', 3, 3) vaut 'ple'

si arg3 est absent, l'extraction commence à partir du début : substr('apples',3) vaut 'app'

Expressions et assignations

Exemples de traitement des dates :

- today() renvoie la date du jour dans le format de SAS (nombres de jours écoulés depuis le 1er Janvier 1960)

Cette fonction renvoyait la valeur 19256 le 20 septembre 2012

Exemples de Statistiques descriptives :

- std(arg1, arg2, arg3,….) où arg1, arg2, arg3 sont des valeurs numériques calcule l'écart-type

std(1,2,3) produit le résultat : 1

- range (arg1, arg2,arg3,…) où arg1, arg2, arg3 sont des valeurs numériques donne l'étendue de la série

range(11, -4, 3) donne le résultat 15

Expressions et assignations

Exemples de mathématiques, probabilités et trigonométrie :

- abs(arg) où arg est une valeur numérique : renvoie la valeur absolue de arg
- log10(arg) est une valeur numérique positive : calcule le logarithme en base 10 de arg
- mod (arg1, arg2) où arg1 et arg2 sont des valeurs numériques : donne le reste de la division de arg1 par arg2
- probnorm(arg) où arg est une valeur numérique : calcule la probabilité que z (Normale standardisée) soit inférieur ou égal à arg

Exemple de génération de nombres au hasard

- ranuni() génère un nombre d'une distribution uniforme sur l'intervalle (0,1)

Expressions et assignations

- Arguments-expressions

A condition de respecter le type, une expression peut-être un argument d'une fonction

- Fonction de fonction

On entend par là la possibilité de trouver une fonction parmi les arguments d'une autre.

Ex. :

day (today()) : la fonction today() est argument de la fonction day

- Respect du type

Les arguments passés à une fonction doivent correspondre au type permis par la fonction

Ex. :

std ('b', 3 ,4) provoquera une erreur (argument non-numérique)

- **LISTE DES FONCTIONS : Voir « Aide » - Introduction au système SAS**

Expressions et assignations

- **Les OPERATEURS sont des traitements appliqués aux opérandes**

Mis à part quelques opérateurs spécifiques (les signes – et + notamment), tous les opérateurs s'appliquent à deux opérandes ; ces opérateurs, appelés infixes, incluent :

1. Opérateurs arithmétiques :

Symbole	Definition
+	addition
/	division
**	Exposant /puissance
*	multiplication
-	soustraction

- **pas d'espace dans les opérateurs composés de deux caractères**

Expressions et assignations

2. Opérateurs de comparaison ou opérateurs relationnels (directement importé de l'aide SAS) :

Symbole	Mnemonic Equivalent	Définition
=	EQ	equal to
^=	NE	not equal to
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal to
<=	LE	less than or equal to
<>		maximum
><		minimum
||		concatenation
	IN	equal to one item in a list

- **pas d'espace dans les opérateurs composés de deux caractères**

Expressions et assignations

Opérateurs de comparaison ou opérateurs relationnels (suite) :

- Le résultat est 1 si la comparaison est Vraie, 0 si elle est Fausse.
- La comparaison des chaînes de caractères est réalisée en commençant par la position la plus à gauche sur base de la valeur en ASCII (du moins pour Windows).
- Lors d'une comparaison numérique, une valeur manquante est considérée comme plus petite.

Expressions et assignations

3. Opérateurs Logiques : AND OR NOT

Symbol	Mnemonic Equivalent	Definition
&	AND	AND comparison
|	OR	OR comparison
¬ *	NOT	NOT comparison
^ *	NOT	NOT comparison
~ *	NOT	NOT comparison

- **pas d'espace dans les opérateurs composés de deux caractères**

Une expression composée de plusieurs opérateurs est évaluée en tenant compte de règles de priorité entre groupes mais les parenthèses sont évaluées en premier lieu (en commençant par les parenthèses les plus intérieures)

Expressions et assignations

Opérateurs Logiques : AND OR NOT

- opérande1 AND opérande2 : le résultat est 1 (vrai) si les 2 opérandes valent 1 (vrai)
  →sinon le résultat est 0 (faux)

- opérande1 OR opérande2 : le résultat est 1 (vrai) si l'un des opérandes ou les deux opérandes valent 1 (vrai)
  →sinon le résultat est 0 (faux)

- NOT opérande : il s'agit de l'inversion logique
  →le résultat vaut 1 si l'opérande vaut 0
  →le résultat vaut 0 si l'opérande vaut 1

Expressions et assignations

Règles de priorité des opérateurs

Groupe 1	** signes + et – NOT >< <>
Groupe 2	* /
Groupe 3	+ -
Groupe 4	||
Groupe 5	< <= ~= >= > =
Groupe 6	AND
Groupe 7	OR

METTRE DES PARENTHESES ET CA MARCHE

A l'intérieur d'un groupe, l'ordre des opérateurs correspond à l'ordre de priorité (évaluation de gauche à droite, **sauf Groupe 1**)

Expressions et assignations

Exemples d'expressions

Groupe 1	total ** 2 -(a+b) NOT (x)
Groupe 2	quantite * prix police/population
Groupe 3	var1+var2 montant-ristourne
Groupe 4	nom||prenom
Groupe 5	age > 18 nom='Dupont'
Groupe 6	age > 18 and sexe = 'F'
Groupe 7	nom = 'Dupont' OR nom = 'durant'

A l'intérieur d'un groupe, l'ordre des opérateurs correspond à l'ordre de priorité (évaluation de gauche à droite, **sauf Groupe 1**)

Expressions et assignations

Exemples d'assignations

v1= sqrt (100*v2);

nbrpop = nbrpop / popul;

dummy= (population = 1000);

binary = (population > 100000 and policiers > 13000);

vcar= substr ('ABCDEFG',3);

vlong=length(name);

recod=(etud='primaire')*1 + (etud='secondaire inférieur' OR etud='secondaire supérieur')*2 + (etud='autres')*3

! A noter : l'apparition d'une même opérande à gauche et à droite du symbole d'affectation

Expressions et assignations

- L'opérateur d'assignation (=) n'est pas l'opérateur d'égalité du Groupe 5; il affecte la valeur de l'opérande de gauche à l'emplacement de mémoire centrale désigné par la variable à sa gauche.

- Ainsi, x=x+1 signifie que la valeur contenue dans la variable x est incrémentée de 1 et le résultat affecté à x (remplacement de valeur). On comprend aisément que la valeur de x avant l'évaluation de l'expression est perdue.

  Par contre, le résultat du terme (x=x+1) dans une expression donne toujours la valeur 0 (Faux).

  Une expression est évaluée pour toutes les lignes du tableau (nombre d'observations), l'affectation suit la même règle.

Expressions et assignations

Assignation et cumul de valeurs

Le programme:

```
data ;
input x;
x=x+1;
datalines;
10
20
30
;
```

- écrit dans le fichier les valeurs:

11
21
31

Il est souvent utile de procéder au cumul des valeurs d'une variable, dans le cas ci-dessus, on désire procéder au **cumul** des valeurs de x , p.ex. pour calculer la moyenne arithmétique

- À votre avis, dans le programme ci-dessus, remplacer x=x+1 par x=x+x donne quel résultat ?

Expressions et assignations

On définit une nouvelle variable qui s'incrémente de la valeur de la variable qu'on désire cumuler à chaque ligne du fichier de données (sum ci-dessous)

```
data ;
input x;
sum=sum+x;
put sum;          /* PUT écrit le contenu d'une variable dans le Journal */
datalines;
10
20
30
;
```

- Ne donne pas le résultat attendu : la valeur de sum qui est un emplacement en mémoire Ram à la première exécution du data n'est pas connue, donc prend la valeur **.** (valeur manquante) et les cumuls des observations suivantes avec cette valeur donnent toujours le résultat **.**

- Que nous dit SAS, dans le journal : **.**

```
NOTE: Des valeurs manquantes ont été générées à la suite d'une opération sur des valeurs manquantes.
      Chaque endroit est défini par : (Nombre de fois) dans (Ligne):(Colonne).
      3 dans 27:8
```

Expressions et assignations

- L' instruction:

retain var1 valeur-initiale var2 valeur-initiale ;

où var1 et var2 sont les variables de cumul et les valeurs suivantes leurs valeurs de départ

résout la difficulté. Le but est de retenir la valeur des variables d'une itération à l'autre dans le data step et de fixer la valeur initiale avant la première itération.

Expressions et assignations

Le programme suivant donne un résultat correct :

```
data ;
retain sum 0;
input x;
sum=sum+x;
put sum;
datalines;
10
20
30
;
```

- Que contient le fichier dans le programme suivant? (Afficher le contenu du fichier)

```
data;
retain nbre 0;
input nom $;
nbre=nbre+1;
datalines;
Jim
Tom
Paul
;
run;
```

Expressions et assignations

On veut convertir tous les prénoms en majuscules…

```
data ;
 input name $ week1 week2;
 weekmean = (week1 + week2) /2;
name = upcase(name);
   datalines;
Julie 195  163
Audrey 220  198
France 173  155
Judith 135  116
;
proc print ;
run;
```

Les instructions de programmation sont exécutées autant de fois qu'il y a d'observations

Expressions et assignations

Exercice

On revient au fichier des données de l'enquête ESS 2006 (Belgique)

Les variables :

1. TV watching, total time on average weekday (0 No time at all 1Less than 0,5 hour 2 0,5 hour to 1 hour 3 More than 1 hour, up to 1,5 hours 4 More than 1,5 hours, up to 2 hours 5 More than 2 hours, up to 2,5 hours 6 More than 2,5 hours, up to 3 hours 7 More than 3 hours)
2. How interested in politics (1 Very interested 2 Quite interested 3 Hardly interested 4 Not at all interested)
3. How happy are you (0 Extremely unhappy 1 – 10 Extremely happy)
4. Gender (1 Male 2 Female)
5. Year of birth

Le fichier texte à lire pour créer le sas data set s'appelle ess 2006 subset.txt et se trouve sur le disque d: (répertoire racine); voici les premières lignes (observations):

73 921939	**En colonne 1 : les réponses à la question 1**
74 821935	**En colonne 2 : les réponses à la question 2**
72 811978	**En colonnes 3 et 4 : les réponses à la question 3**
53 911961	**En colonne 5: les réponses à la question 4**

QUESTIONS:

- **Créer le fichier de données SAS dans une librairie de nom ESS (sur le disque d:), son nom est ess2006**
- **Générer les variables suivantes:**
  - **age des répondants lors de l'enquête**
  - **une variable bonheur à deux valeurs : 1 si les réponses à la question 3 sont supérieures ou égales à 6 et 2 dans le cas contraire**
  - **une variable interet qui vaut 0 si les réponses à la question 2 sont 1 ou 2 et qui vaut 1 si si les réponses sont 3 ou 4**
  - **calculer la moyenne des âges des répondants à l'enquête**

Expressions et assignations

Corrigé de l'exercice : partie acquisition et mise en forme des données

```
libname ess 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et
2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
/* nb: le dossier ess qui contient la librairie est sur le Bureau à adapter
selon l'ordinateur */
data ess.ess2006;
infile 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes
cours\Partim_I_-_Le_langage_SAS\Data\ess_2006_subset.txt';
/* les données sont lues dans le fichier ess_2006_subset.txt qui se trouve
dans le dossier SAS partim I sur le Bureau à adapter selon l'ordinateur */

input tv 2-2
politics 3-3
happy 4-5
gender 6-6
year_of_birth 7-10 ;

proc print; run;
```

Expressions et assignations : RAPPEL

Exemples d'assignations

v1= sqrt (100*v2);

nbrpop = nbrpop / popul;

dummy= (population = 1000);

binary = (population > 100000 and policiers > 13000);

vcar= substr ('ABCDEFG',3);

vlong=length(name);

recod=(etud='primaire')*1 + (etud='secondaire inférieur OR etud='secondaire supérieur')*2 + (etud='autres')*3

! A noter : l'apparition d'une même opérande à gauche et à droite du symbole d'affectation

Expressions et assignations

Corrigé de l'exercice

```
libname ess 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim 1 et
2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
/* nb: le dossier ess qui contient la librairie est sur le Bureau à adapter
selon l'ordinateur */

data ess.ess2006;
infile 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim 1 et 2\mes
cours\Partim_I_-_Le_langage_SAS\Data\ess_2006_subset.txt';
/* les données sont lues dans le fichier ess_2006_subset.txt qui se trouve dans
le dossier SAS partim I sur le Bureau à adapter selon l'ordinateur */

retain sum_age 0 obs 0;
input tv 2-2
politics 3-3
happy 4-5
gender 6-6
year_of_birth 7-10 ;

bonheur = (happy >= 6) * 1 + (happy < 6)*2;
interet = (politics = 1 or politics = 2) * 0 + (politics = 3 or politics = 4) *
1;

age=2006-year_of_birth;
sum_age = sum_age + age;
obs=obs+1;
moyenne_age = sum_age / obs;

proc print; run;
```

Expressions et assignations

Une alternative au « retain » existe:

```
data ;
input x;
sum+x;
put sum;
datalines;
10
20
30
;
```

Expressions et assignations

Corrigé de l'exercice avec la méthode alternative

```
libname ess 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim 1 et
2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
/* nb: le dossier ess qui contient la librairie est sur le Bureau à adapter
selon l'ordinateur */

data ess.ess2006;
infile 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim 1 et 2\mes
cours\Partim_I_-_Le_langage_SAS\Data\ess_2006_subset.txt';
/* les données sont lues dans le fichier ess_2006_subset.txt qui se trouve dans
le dossier SAS partim I sur le Bureau à adapter selon l'ordinateur */

input tv 2-2
politics 3-3
happy 4-5
gender 6-6
year_of_birth 7-10 ;

bonheur = (happy >= 6) * 1 + (happy < 6)*2;
interet = (politics = 1 or politics = 2) * 0 + (politics = 3 or politics = 4) *
1;

age=2006-year_of_birth;
sum_age + age;
obs+1;
moyenne_age = sum_age / obs;

proc print; run;
```

Chapitre 5

Les structures de contrôle

Les structures de contrôle

Introduction

- Jusqu'ici, le Data step se résume à un ensemble d'instructions à logique séquentielle (les instructions sont exécutées l'une à la suite de l'autre en commençant par « data » et en se terminant par « run ».
- Cette logique prédéterminée ne tient pas compte d'événements qui peuvent se produire pendant l'exécution.

Exemples:

1. Dans une expression, la fonction sqrt(x) calcule la racine carrée des valeurs de la variable x. Si x est négatif, que se passe-t-il?
2. Une expérience a besoin d'un échantillon de 100000 valeurs aléatoires à distribution uniforme, on utilise la fonction ranuni(). Doit-on l'écrire 100000 fois dans le « data step »?

Les structures de contrôle

- Tous les langages de programmation disposent d'instructions spécifiques qui permettent de dépasser la logique séquentielle de base, ces instructions sont appelées les « structures de contrôle du programme »; elles sont réparties en trois groupes:

1. Les instructions qui permettent **l'exécution conditionnelle**;
2. Les instructions qui permettent **la répétition**;
3. Les instructions de **branchement**.

Elles recourent pour la plupart à l'utilisation de plusieurs mots réservés.

1 . l'exécution conditionnelle
if, where, select ...

Les structures de contrôle : l'exécution conditionnelle

- Les instructions de ce groupe permettent de modifier la logique séquentielle d'un programme en permettant l'exécution d'une partie de celui-ci selon qu'une (ou des) condition(s) se réalise(nt).
- L'exécution est déterminée par le résultat (Vrai ou Faux) d'une expression.
- En SAS, deux instructions permettent l'exécution conditionnelle:
  - instruction **IF**
  - instruction **SELECT**

Reprenons notre exemple vu plus haut et essayons le programme suivant :

```
data;
input x;
y=sqrt(x);
datalines;
10
-2
30
;
```

Les structures de contrôle : l'exécution conditionnelle

Dans le journal, on peut lire l'avertissement suivant :

```
NOTE: Argument non valide pour la fonction SQRT dans ligne 3
colonne 3.
REGLE :    ----+----1----+----2----+----3----+----4----+----5----
+----6----+----7----+----8----+----9----+
6          -2
x=-2 y=. _ERROR_=1 _N_=2
NOTE: Les opérations mathématiques n'ont pu être effectuées aux
places suivantes. Valeurs manquantes
      affectées.
```

Toutefois, on remarque que le programme ne s'est pas arrêté et que la dernière racine a été calculée.

Que faire pour éviter à SAS d'essayer de calculer la racine d'un nombre négatif ?

Les structures de contrôle : l'exécution conditionnelle

L'exécution conditionnelle IF

Elle prend deux formes possibles:

1. **IF** expression **THEN** instruction; **ELSE** instruction;
2. **IF** expression **THEN** instruction;

→ Les **points-virgules** sont importants !

et une forme spécifique:

- **IF** expression ;

La première forme correspond à l'alternative : si l'évaluation de l'expression conduit à un résultat VRAI, alors l'instruction qui suit THEN est exécutée.

Si le résultat est FAUX, c'est l' instruction qui suit ELSE qui est exécutée.

Pour autant qu'elle soit exécutable, toute instruction du langage, y compris l'assignation, est permise.

Les structures de contrôle : l'exécution conditionnelle

L'exécution conditionnelle IF

Exemple précédent :

```
data;
input x;
IF x >= 0 then y=sqrt(x);
datalines;
10
-2
30
;
```

Ex.

IF (age > 25 AND sexe = 'F') then vbin = 0; else vbin = 1;

Dans la deuxième forme, on ne s'intéresse qu'à une branche de l'alternative, celle où le résultat de l'expression est VRAI

IF lieu = 'Liège' then lieu = 'LIEGE';

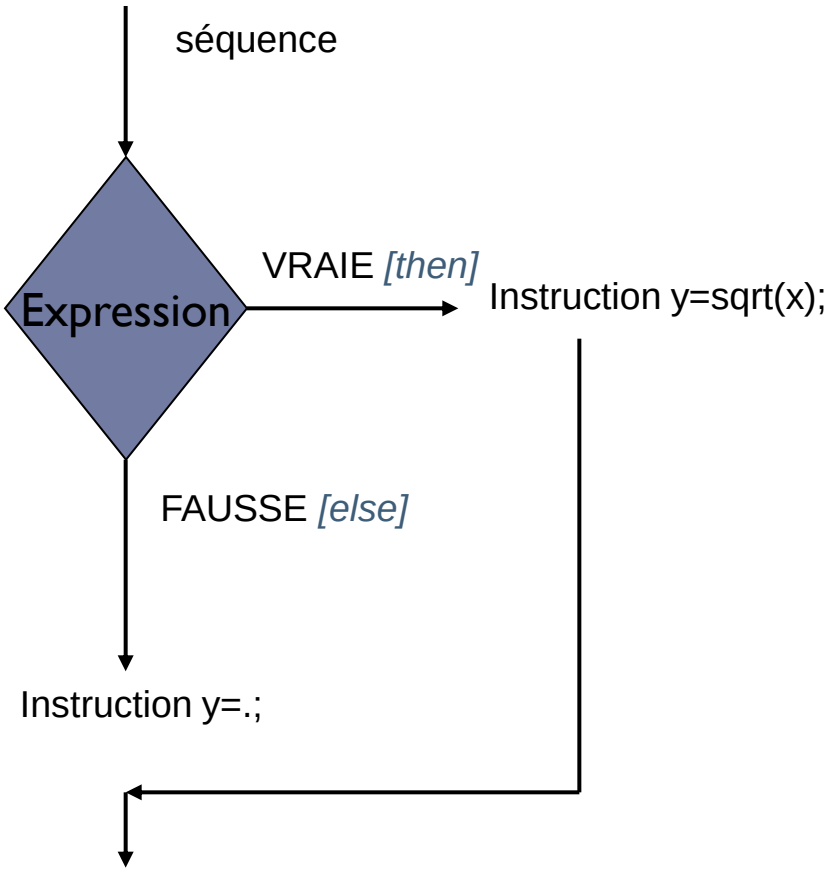
Les structures de contrôle : l'exécution conditionnelle

Exemple de programme contenant une exécution conditionnelle avec « then » et « else »:

```
data;
input x;

If (x gt 0) then y=sqrt(x); else y=. ;
z=x**2;
datalines;
10
-2
30
;
proc print;
run;
```

Organigramme de l'alternative



Les structures de contrôle : l'exécution conditionnelle

Lorsque plusieurs instructions sont contrôlées par IF, on les insère dans un groupe (appelé « groupe do ») au moyen des mots réservés **DO; END;**

Ex.

```
if age <= 25 then do;
categ = '1';
nom = 'Plus petit ou égal à 25 ans';
end;
else do;
categ = '2';
nom = 'Plus grand que 25 ans';
end;
```

Les structures de contrôle : l'exécution conditionnelle

- Les « groupes do » permettent une programmation structurée (rédaction de programmes sous la forme de modules)

- L'instruction contrôlée par IF peut correspondre à un autre IF, on parle d'instructions IF imbriquées.

Ex. if age <= 25 then if sexe = 'F' then x=y/2; else x=y/4; else put age;

if age <= 25 then

if sexe = 'F' then x=y/2; else x=y/4;

else put age;

Les structures de contrôle : l'exécution conditionnelle

3. La troisième forme (forme spécifique) permet la sélection d'un sous-ensemble d'observations (subset) par suppression des observations qui conduisent à un résultat FAUX. Cela revient à exécuter un nombre limité de fois la partie « data step »

Ex.
IF victim = 1;
Dans ce cas, seules les observations pour lesquelles la variable victim vaut 1 sont ajoutées au fichier.

Exemple :

```
data ;
   input name $ week1 week16;
   if (week1 >= 170 and week16 <170) then weekmean = (week1 + week16) /2; else weekmean
= 99;
if name ne 'France';
datalines;
Julie 195  163
Audrey 220  198
France 173  155
Judith 135  116
;
run;
```

- Remarquons que : if name = 'France' then delete ; est équivalent à if name ne 'France';

Les structures de contrôle : l'exécution conditionnelle

La sélection d'observations dans un fichier SAS, à condition qu'il soit déjà créé, peut être réalisée par l'instruction **where** (expression);

Ex.

```
data;
set one;
where x >= 20 ;
run;
```

Le programme suivant est incorrect:

```
data test;
input x;
where x  >= 20;
datalines;
10
20
30
;
```

puisque le fichier test n'est pas créé antérieurement (IF obligatoire)

Les structures de contrôle : l'exécution conditionnelle

/* Sélection des répondants 'Femmes' âgées de moins de 35 ans dans l'enquête européenne
Le fichier a été créé dans la librairie ESS et s'appelle ess2006 CF le programme de création supra
La sélection génère un nouveau fichier dont le nom est femmes dans la même librairie*/

```
libname ess 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
data ess.femmes;
set ess.ess2006;
where (gender=2 and age < 35); /* note : gender=2 pour les femmes */
Run;
```

Forme équivalente du where:

if gender=2 and age <= 35;

Les structures de contrôle : l'exécution conditionnelle

- **L'exécution conditionnelle SELECT**
  Cette instruction permet d'exécuter une (ou plusieurs) instructions selon l'évaluation d'une expression pouvant conduire à plusieurs résultats (logiques, numériques ou de type caractère)

select (expression);
when (*resultat1*) *instruction;*
when (*resultat2*) *instruction;*
when (*resultat3*) *instruction;*
…..
<**otherwise** *instruction;*>
end;
(nb:<….> est une option)

- L'expression renvoie un résultat qui est comparé à resultat1, resultat2, resultat3, …
- En cas d'égalité, l'instruction associée au when est exécutée et le programme continue ensuite après l'instruction end.
- Si l'égalité n'est pas trouvée et si la clause « otherwise » est absente, SAS signale une erreur.

Comme dans le cas de l'instruction IF, lorsque plusieurs instructions sont exécutées, on les insère dans un groupe au moyen des mots réservés
DO; END;

Les structures de contrôle : l'exécution conditionnelle

Exemple 1: expression numérique entière

```
select (a);
when (1) x=x*10;
when (2);
when (3,4,5) x=x*100;
otherwise;
end;
```

Remarque : when(2) est-il nécessaire?

Exemple 2: expression de type texte (caractère)

```
select (payclass);
when ('monthly') amt=salary;
when ('hourly') do;
amt=hrlywage*min(hrs,40); if hrs>40 then put 'CHECK TIMECARD';
end; /* end of do */
otherwise put 'PROBLEM OBSERVATION';
end; /* end of select */
```

(Exemples extraits de la documentation en ligne de SAS)

Les structures de contrôle : l'exécution conditionnelle

Exercice:

Un fichier de type texte a été encodé avec les données suivantes:

(le nom, la taille, le poids et l'âge)

Jean 180 72 25

Tom 172 68 30

Mike 72 6 40

Helene 55 160 25

Après avoir procédé à la création du fichier de données en SAS, on s 'aperçoit qu'il contient les erreurs suivantes:

- La taille de Mike a été encodée en pouces (1 pouce = 2.5cm) et son poids en dizaine de kilos
- On a encodé Helene alors qu'il s'agit de Jane et on a inversé taille et poids

Corrigez ces erreurs dans SAS en utilisant les structures conditionnelles

Les structures de contrôle : l'exécution conditionnelle

- Création du fichier

```
libname ess 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\pourtest\ess\';
/*Pour le choix du nom de librairie cf. supra */
data ess.etud;
/*Le fichier créé s'appelle etud */
      input name $ taille poids age;
datalines;
Jean 180 72 25
Tom 172 68 30
Mike 72 6 40
Helene 55 160 25
   ;
proc print;
run;
```

Les structures de contrôle : l'exécution conditionnelle

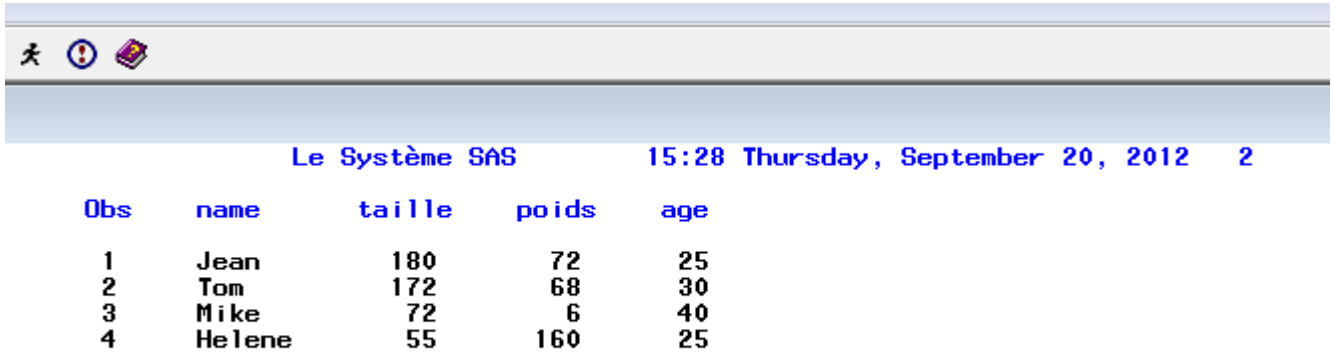
- Correction du fichier

```
/*---------- Correction du fichier ---------------------------------------------*/
data ess.etud2;
set ess.etud;
select(name);
when('Mike') do;              /* Forme équivalente : IF name = 'Mike' then do;  */
  taille=taille*2.5;
  poids=poids*10;
end;
when('Helene') do;          /* Forme équivalente : IF name = 'Helene' then do; */
  name='Jane';
  sauve=taille;
  taille=poids;
  poids=sauve;
  end;
  otherwise;                    /* Si on adopte la Forme équivalente, on supprime cette instruction */
end;
Proc print;
Run;
```

Les structures de contrôle : l'exécution conditionnelle

Le résultat SAS (tel qu'il apparait dans la sortie SAS)

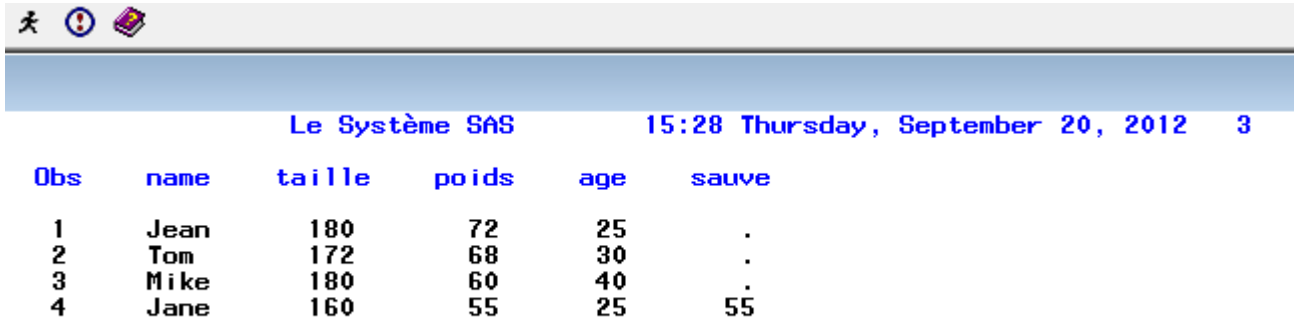
- Après création du fichier



Le Système SAS 15:28 Thursday, September 20, 2012 2

Obs	name	taille	poids	age
1	Jean	180	72	25
2	Tom	172	68	30
3	Mike	72	6	40
4	Helene	55	160	25

- Après correction du fichier



Le Système SAS 15:28 Thursday, September 20, 2012 3

Obs	name	taille	poids	age	sauve
1	Jean	180	72	25	.
2	Tom	172	68	30	.
3	Mike	180	60	40	.
4	Jane	160	55	25	55

Les structures de contrôle : l'exécution conditionnelle

Exercice

On revient au fichier de données ESS pour la Belgique (cf. exercices précédents);

-la variable âge doit être catégorisée :

- moins de 25 ans
- de 25 à 45
- de 45 à 60
- plus de 60

on établit le comptage des répondants selon les catégories créées

-on se demande si les personnes qui se disent « Heureuses » (réponse > 6) ne sont pas plus jeunes.

Corrigé

```
/* On utilise le fichier créé précédemment dans la lirairie ess
Le fichier s'appelle ess2006
On crée un nouveau fichier dont le nom est exercice dans la même librairie
*/

libname ess 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes
cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
data ess.exercice;
set ess.ess2006;

age=2006-year_of_birth;
/* Création de groupes d'ages */
if (age < 25) then gr_age = 1;
if (age >= 25 and age < 45 ) then gr_age = 2;
if (age >= 45 and age < 60 ) then gr_age = 3;
if (age >= 60) then gr_age = 4;
/* Comptages des modalités */
select(gr_age);
when(1) gr1+1;
when(2) gr2+1;
when(3) gr3+1;
when(4) gr4+1;
end;
run; /* A supprimer si on écrit le programme d'un seul coup*/
```

```
/* Création de la variable bonheur*/
bonheur = (happy< 6)* 2 + (happy >= 6) * 1;
/* Moyenne d'ages selon les deux catégories */
select(bonheur);
when (1) do;
tot1+age;
n1+1;
mean1=(tot1/n1);
end;
when (2) do;
tot2+age;
n2+1;
mean2=(tot2/n2);
end;
end;
proc print;
run;
/* NB la création des groupes d'ages peut se faire par
select ;
when(age < 25)  gr_age = 1;
when (age >= 25 and age < 45 )  gr_age = 2;
when (age >= 45 and age < 60 )  gr_age = 3;
when (age >= 60)  gr_age = 4;
end;
/*
```

2. l’exécution répétitive

Les structures de contrôle : l'exécution répétitive

L'exécution répétitive : la boucle

Cette structure de contrôle permet de ***répéter un certain nombre de fois une partie du programme.***

Elle prend trois formes possibles, dont les deux premières sont les plus couramment utilisées:

1. **DO** itératif **;**
2. **DO WHILE** ;
3. **DO UNTIL;**

Les structures de contrôle : l'exécution répétitive

- **Première forme : DO itératif**

DO *variable*=*start* ***TO*** *stop* <***BY*** *increment*>;

Instructions…;

END;

(nb:<….> est une option)

L' ensemble des instructions comprises entre le mot **do** et le mot **end** constitue la partie du programme soumise à l 'exécution répétitive, on l'appelle le corps de la **boucle**.

- *variable est une variable SAS (**le compteur**) de type numérique (on se limitera aux v. entières)*
- *start est la valeur de **début du compteur***
- *stop est la **valeur finale du compteur***
- *increment est la **constante ajoutée au compteur à chaque itération** (par défaut : valeur 1)*

Les structures de contrôle : l'exécution répétitive

Fonctionnement :

- A l'entrée dans la boucle, la variable compteur s'initialise à la valeur **start** et les instructions du corps de boucle sont exécutées.
- Lorsque le programme arrive à la fin de la boucle (instruction **end**), il revient à l'instruction **DO**, la variable compteur est incrémentée de incrément;
  Si elle reste inférieure ou égale à la valeur **stop**, les instructions du corps de boucle sont ré-exécutées.
- La répétition se termine lorsque la valeur de la variable compteur est supérieure à la valeur **stop**.
  Le programme se poursuit à l'instruction suivant l'instruction **end**.

- A l'entrée dans la boucle, si la valeur **start** est supérieure à la valeur **stop**, le programme quitte la structure et se poursuit à l'instruction suivant l'instruction **end**.

Les valeurs de départ, d'arrêt et d'incrémentation peuvent être des expressions mais doivent être numériques (le plus souvent positives et entières).

L'incrément peut être négatif; dans ce cas, la valeur de **start** est évidemment supérieure à la valeur **stop**.

Les structures de contrôle : l'exécution répétitive

```
/* génération de 20 nombres au hasard entre 0 et 1 (non compris) affichés
dans le Journal
nb: l'instruction PUT v1 v2 …vn écrit le contenu des variables v1 v2 …vn
dans le journal */
data;
do i=1 to 20;


        x=uniform(0);
/* Voir la description de cette fonction dans l'Aide */
    put i x;
/* Ecrit dans le Journal les valeurs de la variable compteur et le résultat
de la fonction*/
end;
run;
```

Remarques:

- Le Journal affiche les 20 valeurs générées bien que le fichier de données en SAS ne contienne qu'une observation

Explication: les lignes sont écrites à la fin du data step

Les structures de contrôle : l'exécution répétitive

- L'instruction **output** permet de créer une ligne du fichier à chaque itération.

```
/* génération d'un jet de 1000 fois un dé et calcul de la moyenne
arithmétique */
data;
do i=1 to 1000;
dice=int(1+6*(uniform(0)));
output;
end;
run;
```

Les structures de contrôle : l'exécution répétitive

Comment être sûr que le dé n'est pas pipé?

```
/* Génération des résultats de 1000 jets d'un dé - Comptage des valeurs 1-6 et moyenne arithmétique */
data;
retain total 0;
retain moyenne 0;

do i=1 to 1000;
dice=int(1+6*(uniform(0)));          /* Simulation jet de dé */
select (dice);
            when (1) somme1 + 1; /* Comptage de la face 1  */
            when (2) somme2 + 1; /* Comptage de la face 2  */
            when (3) somme3 + 1;/*  etc…                                */
                  when (4) somme4 + 1;
                  when (5) somme5 + 1;
                  when (6) somme6 + 1;
            end;
total = total + dice; /* Somme des résultats des tirages  */
moyenne = (total/i); /* La moyenne arithmétique    */
output;
end;
proc print;
run;
```

Les structures de contrôle : l'exécution répétitive

Deuxième forme : DO WHILE *[tant que]*

DO while (expression) ;
Instructions…;
END;
Fonctionnement:

- A l'entrée dans la boucle, si expression est Vraie, les instructions sont exécutées jusqu'à **END**. Le programme recommence à l'instruction **DO**, évalue l'expression;
- Si elle est Vraie les instructions sont exécutées jusqu'à **END**.
- Si l'expression est Fausse, le programme se poursuit à l'instruction qui suit le **END.**

Troisième forme : DO UNTIL *[jusqu'à ce que]*

DO until (expression) ;
Instructions…;
END;
Les instructions sont exécutées itérativement jusqu'à ce que l'expression entre parenthèses, évaluée au END, soit vraie .

Les structures de contrôle : l'exécution répétitive

Comparer les résultats des deux programmes suivants:

```
data;
n=6;
do until (n>5);
put n;
end;
Run;
```

VS

```
data;
n=6;
do while (n>5);          /* à votre avis, que va donner ce prog?*/
  put n;
end;
run;
```

Les structures de contrôle : l'exécution répétitive

Quelques remarques:

1. Dans les boucles WHILE et UNTIL, le nombre de répétitions n'est pas connu lorsque le programme entre dans la boucle, à l'inverse du DO itératif
2. Dans DO WHILE, l'expression doit devenir fausse pour éviter un bouclage infini.
3. Dans DO UNTIL, l'expression doit devenir vraie.
4. Si expression n'est pas logique (prend des valeurs non limitées à 0 ou 1),
    1. Elle est fausse si elle est nulle ou missing ;
    2. Elle est vraie dans tous les autres cas.
5. La boucle DO UNTIL est exécutée au moins une fois.
6. L'utilisation des compteurs permet de modifier le résultat de l'expression.

Les structures de contrôle : l'exécution répétitive

Exemple

```
data;
n=0;
do while (n<5);
  put n;
  n+1;
end;
run;
```

- L'instruction d'accumulation génère automatiquement une instruction RETAIN sur la variable utilisée comme compteur et l'initialise à zéro.

Les structures de contrôle : l'exécution répétitive

- A partir d'une table de fréquences croisées, on peut reconstituer le fichier des données de base en utilisant une boucle DO ITERATIF,
- Ensuite on désire sélectionner les victimes.

	Victimes (1)	Non-Victimes (2)
<25 ans (1)	25	9
25-60 (2)	100	96
> 60 ans(3)	36	50

```
/*      DO itératif     */
Data fichier1;
       input categ_age victim count;
  do i = 1 to count;
      output ;
  end;
datalines;
   1 1 25
   1 2 9
   2 1 100
   2 2 96
   3 2 50
   3 1 36
   ;
Data;
Set fichier1;
If victim=1;
proc print;
run;
```

Les structures de contrôle : l'exécution répétitive

- A partir d'une table de fréquences croisées, on peut reconstituer le fichier des données de base en utilisant une boucle DO WHILE,
- Ensuite on désire sélectionner les victimes.

	Victimes (1)	Non-Victimes (2)
<25 ans (1)	25	9
25-60 (2)	100	96
> 60 ans(3)	36	50

```
/*      DO while      */
Data fichier2;
      input categ_age victim count;
compteur=1;
   do  while (compteur <= count);
              output ;
              compteur+1;
    end;
datalines;
   1 1 25
   1 2 9
   2 1 100
   2 2 96
   3 2 50
   3 1 36
   ;
Data selection; set fichier2; if victim=1;
proc print;
run;
```

3. les instructions de branchement

Les structures de contrôle : les instructions de branchement *(pour mémoire)*

- Ces instructions permettent d'exécuter des instructions en-dehors de la séquence initiale.

- Instruction : **GOTO** étiquette;

où étiquette est un nom SAS qui sert de référence à une instruction du programme ; la partie « étiquette » est séparée de la partie « instruction » par « : »

Cette instruction de branchement est souvent (sinon toujours) associée à la structure de contrôle d'exécution conditionnelle.

- instruction : **LINK** étiquette;

L'exécution se poursuit à l'endroit référencé par l 'étiquette jusqu'à la rencontre d'une instruction RETURN; le programme revient à l'instruction qui suit LINK. (sous-programme)

Les structures de contrôle : les instructions de branchement *(exemple venant de l'aide SAS)*

- Instruction : **GOTO** étiquette;

jumps to a new statement

GOTO label**;**

where *label* is a labeled statement. Execution jumps to this statement. A label is a name followed by a colon (:).

GOTO statements are usually clauses of IF statements, for example,

```
if x>y then goto skip;
y=log(y-x);
yy=y-20;
skip: if y<0 then
   do;
     more statements
   end;
```

Chapitre 6

Gestion des fichiers de données en SAS (suite)

Gestion des fichiers de données en SAS (suite)

Instructions additionnelles pour la mise à jour de fichiers

- **delete ;** forme équivalente : **remove;**
  exclut l'observation en cours du fichier
- **keep** *v1 v2 v3 …;*
  enregistre les colonnes (variables) dont les noms figurent dans la liste
- **drop** *v1 v2 v3 …;*
  supprime les colonnes (variables) dont les noms figurent dans la liste
- **rename** *v1=v1new v2=v2new …..;*
  modifie le nom des colonnes (variables)
- **label** *v1 = 'descriptif var1' v2 = 'descriptif var2' …. ;*
  associe des étiquettes aux variables (max. 256 caractères)

Gestion des fichiers de données en SAS (suite)

Ex.

```
Data gestion_d;
input prenom $ no $  a  ;
if no = 'Jones' then delete;
nom = upcase (no);
label a = 'Age de la personne';
rename a = age;
drop no;     /* on pourrait écrire : keep nom prenom age*/
datalines;
Brad Pitt 22
Tom Jones 25
Richard Geere 30
Tom Hanks 40
;
proc print label;
run;
```

EXERCICE

Dans l'enquête ESS réalisée en 2008, je souhaite comparer les réponses à la question « Etes-vous heureux » des participants belges et français.

Pour comparer, je retiens deux calculs:

- **La moyenne arithmétique des réponses dans les deux pays**
- **Le pourcentage des réponses au-dessus de la valeur 6 pour les deux pays.**

Ecrivez le programme sas qui crée le fichier des données à partir des données extraites sur le site (fichier de type texte) et ensuite réalisez les calculs.

```
libname ess 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\pourtest\ess' ;
/* nb: le dossier ess qui contient la librairie est à adapter selon l'ordinateur */
data ess.happyness;
infile 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\Data\happyness_compared_in_BE_and_FR.txt';

input country $ 1-2 happy 3-4 ;


…
```

Tip :

```
nbe+1;
 sum_happyBE+happy;
 mean_happyBE=sum_happyBE/nbe;
 if (happy>6) then do;
              nb_happyBE+1;
              pctBE=(nb_happyBE/nbe)*100;
            end;
             keep country mean_happyBE pctBE;
             output;
```

```
libname ess 'C:\Users\SEBA Fontaine\Documents\dropbox\mes
cours\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\pourtest\ess\' ;
/* nb: le dossier ess qui contient la librairie est à
adapter selon l'ordinateur */
data ess.happyness;
infile 'C:\Users\SEBA Fontaine\Documents\dropbox\mes
cours\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\Data\happyness_compared_in_BE_and_FR.txt';

input country $ 1-2 happy 3-4 ;
select (country);
when('BE') do;
  nbe+1;
  sum_happyBE+happy;
  mean_happyBE=sum_happyBE/nbe;
  if (happy>6) then do;
                 nb_happyBE+1;
                 pctBE=(nb_happyBE/nbe)*100;
               end;
       keep country mean_happyBE pctBE;
       output;
end;
```

```
when('FR') do;
  nfr+1;
  sum_happyFR+happy;
  mean_happyFR=sum_happyFR/nfr;
  if (happy>6) then do;
                  nb_happyFR+1;
                  pctFR=(nb_happyFR/nfr)*100;
               end;
       keep country mean_happyFR pctFR;
       output;
  end;
otherwise;
end;
proc print;
run;
```

Gestion des fichiers de données en SAS

- Sélection d'observations au hasard

```
/* ce programme lit 100 lignes au hasard dans le fichier ess2006 qui contient
1798 observations*/
/* Pour apprendre à utiliser l'option « point », nous utiliserons l'aide.*/


libname ess 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes
cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
data ess.random;
do i = 1 to 100;
choice=int(1+1798*uniform(0));
set ess.ess2006 point=choice;

output;
if i ge 100 then stop;    /*  !!!!!!!!!!!!!!!!!!!!!!*/
end;
proc print;
run;
```

Gestion des fichiers de données en SAS

- POINT=variable

specifies a temporary variable whose numeric value determines which observation is read. POINT= causes the SET statement to use random (direct) access to read a SAS data set.

- **Requirement:** a STOP statement
- **Restriction**: You cannot use POINT= with a BY statement, a WHERE statement, or a WHERE= data set option. In addition, you cannot use it with transport format data sets, data sets in sequential format on tape or disk, and SAS/ACCESS views or the SQL procedure views that read data from external files.
- **Restriction:** You cannot use POINT= with KEY=.
- **Tip:** You must supply the values of the POINT= variable. For example, you can use the POINT= variable as the index variable in some form of the DO statement.
- **Tip:** The POINT= variable is available anywhere in the DATA step, but it is not added to any new SAS data set.

- **CAUTION :**

Continuous loops can occur when you use the POINT= option.

When you use the POINT= option, you must include a STOP statement to stop DATA step processing, programming logic that checks for an invalid value of the POINT= variable, or both. Because POINT= reads only those observations that are specified in the DO statement, SAS cannot read an end-of-file indicator as it would if the file were being read sequentially. Because reading an end-of-file indicator ends a DATA step automatically, failure to substitute another means of ending the DATA step when you use POINT= can cause the DATA step to go into a continuous loop. If SAS reads an invalid value of the POINT= variable, it sets the automatic variable _ERROR_ to 1. Use this information to check for conditions that cause continuous DO-loop processing, or include a STOP statement at the end of the DATA step, or both

Gestion des fichiers de données en SAS (suite)

- **Création de plusieurs fichiers à partir d'une ligne de données**

Par exemple, le programme :

```
data fem hom;
      input gender notes;
if gender = 1 then output hom;
if gender = 2 then output fem;
datalines;
   1  2
   1  9
   2  10
   1  9
   2  15
   1  18
   ;
```

- crée deux fichiers : hom (si la variable gender vaut 1) et fem (si gender vaut 2)
- Les fichiers peuvent évidemment constituer des membres d'une librairie : data ess.fem ess.hom;

Chapitre 7

Les procédures

Les procédures : procedure step

- Structure générale d'une PROC

PROC nom de procédure options de procédure;

instructions de procédure;

- Le nom de procédure correspond au traitement qu'on désire réaliser sur le fichier de données SAS
- Les options de procédure (mot-clé, argument) définissent l'environnement des données et des résultats.

Ex. : nom du ficher SAS qui contient les données (data = *sas data set*)

- Les instructions de procédure définissent les traitements spécifiques que la procédure doit exécuter.

EX. : limiter la portée de la procédure à certaines variables

Aussi : des instruction globales (cf. infra)

Exemple: Voir documentation on-line

- **Base Procedures**:
- APPEND
- CALENDAR
- CATALOG
- CHART
- CIMPORT
- COMPARE
- CONTENTS
- COPY
- CORR
- CPORT
- DATASETS
- DBCSTAB
- DISPLAY
- EXPLODE
- EXPORT
- FORMAT
- FORMS
- FREQ
- IMPORT
- MDDB
- MEANS
- OPTIONS
- OPTLOAD
- OPTSAVE
- PLOT
- PMENU
- PRINT
- PRINTTO
- PRTDEF
- RANK
- REGISTRY
- REPORT
- SORT
- SQL
- STANDARD
- SUMMARY
- TABULATE
- TIMEPLOT
- TRANSPOSE
- TRANTAB
- UNIVARIATE

- **Product-specific procedures for:**
- SAS/ACCESS
- SAS/AF
- SAS/CONNECT
- SAS/ETS
- SAS/FSP
- SAS/GIS
- SAS/GRAPH
- SAS/IML
- SAS/INSIGHT
- SAS/MDDB Server
- SAS/OR
- SAS/QC
- SAS/SHARE
- SAS/STAT

Ex. PROC PRINT

PROC PRINT <*option(s)*>;

BY <DESCENDING> *variable-1* <...<DESCENDING> *variable-n*>
<NOTSORTED>;

PAGEBY *BY-variable*;

SUMBY *BY-variable*;

ID *variable(s)* <*option*>;

SUM *variable(s)* <*option*>;

VAR *variable(s)* <*option*>;

Voir Aide pour les options de PROC et la syntaxe

Les procédures : structure d'un programme SAS

- **instructions globales;**

DATA;

- **instructions de programmation;**
- **instructions globales;**

PROC …. options;

- **instructions de procédure;**
- **instructions globales;**

PROC …. options;

- **instructions de procédure;**
- **instructions globales;**

Les procédures : structure d'un programme SAS

- **Les instructions globales**

Il s'agit d'instructions qui permettent de transmettre des informations vers le système SAS et sont interprétées par celui-ci dans la phase de compilation.

Elles se répartissent en plusieurs catégories:

- Instructions d'accès aux données

  Ex. filename et libname

- Gestion du Journal

  page; permet de passer à la page suivante

  skip n ; crée n lignes vides

- Gestion de la fenêtre des résultats

  footnote 'texte' ; crée une note de bas de page initialisée à 'texte'

  title <n> 'texte' ; crée un titre en haut de chaque page de résultat en commençant à la ligne n

- Contrôle du programme

  endsas; arrête l'exécution de sas à la fin du data step ou de la proc en cours

  options ….; modifie les options par défaut (options nodate; supprime l'affichage de la date dans les résultats)

- etc…

Les procédures : exemples de procédures

/* Première étape */

```
data surv.fichier;
infile 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\Data\ess_2006_subset.txt';
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\DOC CLEO\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
input  tv 2 politics 3 happyness 4-5 gender 6 birth 7-10;
```

Les procédures : exemples de procédures

```
/* Deuxième étape */
data surv.fichier2;
set surv.fichier;
/* J'aimerais bien un beau titre au début de mon output*/
age=2006-birth;
bonheur = (happyness < 6)* 2 + (happyness >= 6) * 1;
int_pol= (politics = 1 or politics = 2)*0 + (politics = 3 or politics =
4)*1;
label age = 'Age du répondant';
label int_pol = 'Intérêt pour la politique';
/* Il faudrait aussi ajouter une note de bas de page qui dit ceci: « Waw,
une note de bas de page, je ne l'avais pas vue!*/
proc print;
run;
```

Solution :

```
data surv.fichier;
infile 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\Data\ess_2006_subset.txt';
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
input  tv 2 politics 3 happyness 4-5 gender 6 birth 7-10;
data surv.fichier2;
set surv.fichier;
title 'Quel beau titre !';
age=2006-birth;
bonheur = (happyness < 6)* 2 + (happyness >= 6) * 1;
int_pol= (politics = 1 or politics = 2)*0 + (politics = 3 or politics = 4)*1;
label age = 'Age du répondant';
label int_pol = 'Intérêt pour la politique';
footnote 'Waw, une note de bas de page, je ne l''avais pas vue!';
proc print;
run;
```

PROC PRINT <option(s)>; [helpfile SAS]

To do this		Use this option
Specify text for the HTML contents link to the output		CONTENTS=
Specify the input data set		DATA=
Control general format		
	Write a blank line between observations	DOUBLE
	Print the number of observations in the data set, in BY groups, or both, and specify explanatory text to print with the number	N=
	Suppress the column in the output that identifies each observation by number	NOOBS
	Specify a column header for the column that identifies each observation by number	OBS=
	Round unformatted numeric values to two decimal places	ROUND
Control page format		
	Format the rows on a page	ROWS=
	Use each variable's formatted width as its column width on all pages	WIDTH=UNIFORM
Control column format		
	Control the orientation of the column headings	HEADING=
	Use variables' labels as column headings	LABEL or SPLIT=
	Specify the split character, which controls line breaks in column headings	SPLIT=
	Specify one or more style elements for the Output Delivery System to use for different parts of the report	STYLE
	Determine the column width for each variable	WIDTH=

Les procédures : exemples de procédures

- Imprimer un fichier d'une librairie avec options de procédure

```
/* On adapte l'instruction global qui suit selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc print data = surv.fichier2 double noobs round label;
run;
```

- Imprimer un fichier d'une librairie avec options, instructions de procédure et globales

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
title 'Enquête ESS 2006 - Résultats';
options nodate;
proc print data = surv.fichier2 double noobs round label;
  var bonheur age;
  sum age;
run;
```

Les procédures : exemples de procédures

- Aide en ligne : PROC SORT

Ordonne un fichier de données selon les valeurs d'une variable (numérique ou texte) selon un ordre croissant ou décroissant.

Cela est notamment très utile dans le cas d'un « merge » entre deux fichiers,

- EXEMPLE

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age; /* Allons voir dans l'aide les arguments de
"by"*/
run;
```

Les procédures : exemples de procédures

- Aide en ligne : PROC MEANS

(nb: équivalent de SUMMARY)

- Fournit les indices synthétiques de base de la statistique descriptive (variables quantitatives continues)

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age;
run;
proc means data = surv.fichier2;
run;
```

Les procédures : exemples de procédures

- Avec options de PROC pour les calculs statistiques et instructions de proc

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes
cours\SAS partim 1 et 2\mes cours\Partim_I_-
_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age;
run;
proc means data=surv.fichier2  RANGE  SKEWNESS CV STD  SUM MAX
MEAN MIN  MEDIAN Q3 P1 P90 P5 P95 P10 P99 Q1 ;
var age;
run;
```

Les procédures : exemples de procédures

- Idem + traitement par groupe (gender) et enregistrement des résultats

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age;
run;
proc sort data=surv.fichier2;
by gender;
proc means data=surv.fichier2  RANGE  SKEWNESS CV STD  SUM MAX MEAN MIN
MEDIAN Q3 P1 P90 P5 P95 P10 P99 Q1 ;
var age;
by gender;
output out=surv.results;
Proc contents data=surv.results;
Proc print data=surv.results;
run;
```

Les procédures : exemples de procédures

- Aide en ligne : PROC FREQ

fournit les indices synthétiques de base de la statistique descriptive (variables qualitatives)

- principale instruction de procédure:

TABLES var1 var 2…; (comptage des fréquences pour une variable)

TABLES var1*var2 …; (tableaux croisés)

EXEMPLE :

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age;
run;
proc freq data=surv.fichier2 ;
tables gender int_pol;
tables gender*int_pol / chisq ; /* Freq abs, freq rel global, freq rel
ligne, freq rel colone*/
run;
```

Les procédures : exemples de procédures

- Aide en ligne : PROC UNIVARIATE

fournit les indices synthétiques approfondis

EXEMPLE :

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age;
run;
proc univariate data=surv.fichier2 ;
var age;
histogram;
run;
```

Voir la fenêtre graphique dans l'explorateur des résultats + les outputs (très riches)!

Les procédures : exemples de procédures

- Aide en ligne : PROC CHART

permet la représentation des variables qualitatives sous la forme des diagrammes en barres et « quartiers de tartes » (pie chart)

instructions de procédure principales:

vbar var1 var2 …/ options;

hbar idem ..;

pie idem…;

EXEMPLE :

```
/* On adapte l'instruction suivante selon son PC'*/
libname surv 'C:\Users\SEBA Fontaine\Documents\dropbox\mes cours\SAS partim
1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess\';
proc sort data=surv.fichier2 out=surv.result;
by gender descending age;
run;
proc chart data=surv.fichier2 ;
vbar happyness / discrete; /* barres verticales; on peut utiliser
symbol="#" (pour changer le remplissage des barres*/
run;
```

Exercice Récapitulatif

Exercice récapitulatif

Le fichier subset_ess2010.txt (cf. plateforme du cours) est une extraction des données de l'enquête European Social Survey. Il contient les données des répondants BELGES concernant le nombre d'années d'études réalisées selon le sexe.

1. Vous commencez par créer le fichier en format SAS et dans une librairie
2. Vous vérifiez que ces données ne contiennent pas de valeurs anormales et au besoin vous corrigez
3. Par programmation, vous calculez la moyenne arithmétique du nombre d'années d'études selon le sexe
4. Vous sélectionnez au hasard 200 répondants, vous recalculez comme au point 3 et vous comparez les résultats
5. Vous calculez ensuite pour tous les répondants les statistiques descriptives classiques en utilisant les PROC adéquates.

Exercice récapitulatif

SOLUTION

```
libname exos 'C:\Users\LSB\Documents\Dropbox\mes cours\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\pourtest\ess';
data exos.yearsofstudy;
infile
'C:\Users\LSB\Documents\Dropbox\mes cours\SAS partim 1 et 2\mes cours\Partim_I_-_Le_langage_SAS\Data\subset_ess2010.txt';
input gender  1 years 2-3;
if years=88 then years=.;
proc means;
proc freq; tables gender;
run;
data ;
set exos.yearsofstudy;
if gender = 1  then do;
          if years ne . then do;
          sumyears_h+years;
                    nbh+1;
                    meanyearsH=sumyears_h/nbh;
                              end;
               end;
               else do;
                    if years ne . then do;
          sumyears_f+years;
                    nbf+1;
                    meanyearsF=sumyears_f/nbf;
                              end;
               end;
proc print;
proc sort; by gender;
proc means mean;
by gender; var years;
run;
```

```
data exos.random;
do i = 1 to 200;
choice=int(1+1704*uniform(0));
nombre=choice;
set exos.yearsofstudy point=choice;

output;
end;
if i ge 200 then stop;
run;

data ;
set exos.random;
if gender = 1  then do;
          if years ne . then do;
          sumyears_h+years;
                  nbh+1;
                  meanyearsH=sumyears_h/nbh;
                          end;
             end;
              else do;
                  if years ne . then do;
          sumyears_f+years;
                  nbf+1;
                  meanyearsF=sumyears_f/nbf;
                          end;
             end;
proc print;
proc sort; by gender;
proc means mean;
by gender; var years;
run;
```