SAS Base

Support de cours

STID 2ème année 2006-2007
Christelle DEFOSSEZ

Objectifs du cours

- Acquérir les bases de programmation du langage SAS
  - Lecture des données
    - Import et exportation de données
  - Manipulation et transformation de données
    - Agrégation de données
    - Fusion de tables
  - Exploration et analyse des données
    - Procédure statistique

Sommaire

Partie 1

Familiarisation avec l'environnement SAS

1. Le logiciel SAS
2. Les fenêtres SAS
3. Les tables SAS
4. L'espace de stockage
5. Les variables
6. Les types de traitement
7. Options générales de SAS
8. L'instruction DM

Logiciel SAS

- SAS (Statistical Analysis System) est un progiciel, qui à l'origine a été développé pour le calcul statistique. SAS est un progiciel constitué:
  - d'un langage de procédure permettant l'étude des données individuelles,
  - d'un langage de programmation évolué permettant de manipuler et de gérer des fichiers de données.

- SAS en chiffre :
  - Fondée en 1976, 1,2 milliard de dollars en 2001 ,
  - 8 400 employés dans le monde (270 en France)
  - 38 000 sites clients répartis dans 110 pays
  - 3,5 millions d'utilisateurs
  - 30% du chiffre d'affaires annuel réinvesti en R&D

- Actuellement en version 9.

Logiciel SAS

- SAS est composé de plusieurs modules:
  - SAS/BASE: Gestion de données (transformations de données, création de nouvelles variables, tris, sélections, découpe en sous-fichiers, fusion de plusieurs fichiers, ...).
  - SAS/STAT: procédure d'analyse statistique.
  - SAS/ETS: Procédures d'analyse économétrique.
  - SAS/OR: Procédures de recherche opérationnelle.
  - SAS/IML: Langage matriciel interactif.
  - SAS/GRAPH: Procédures graphiques pour tables traçantes et terminaux graphiques.
  - SAS/FSP: Procédures interactives pour la gestion des données.
  - SAS/CONNECT: Traitement coopératif entre sessions SAS.
  - SAS/Integration Technologies: Support pour les clients
  - SAS/Enterprise Guide.

Les fenêtres principales

- Editor : écriture du programme
  - Comprend les foncionnalités d'un éditeur de texte :
    - copier/couper/coller
    - rechercher/remplacer
  - Depuis la Version 8, les différents éléments du programme sont en couleur pour faciliter la lecture :
    - bleu foncé = mots clés
    - vert=commentaire
    - rouge=erreur
    - noir= instruction...
  - Possibilité de réduire – ou déployer + les “étapes” d'un programme
  - Exécution du programme

Les fenêtres principales

- Log : visualisation du déroulement du programme et les erreurs
  - A chaque exécution de programme, le code source est réécrit dans cette fenêtre et chaque étape est commentée.

  - Comment lire une log?
    - **Note** : informations générales sur l'éxécution de chaque étape
      - Temps de traitement (elapsed : temps réel, CPU : temps machine)
      - Nombre d'observations en lecture et écriture
      - Informations propres au code source utilisé...
    - **Warning** : SAS rencontre une erreur qu'il réussit à gérer SEUL, cela correspond-t-il à ce que vous désirez?
    - **ERROR** : Message d'erreur bloquante

Les fenêtres principales

- Output : Visualisation des résultats
  - Receuil des résultats produit suite à l'exécution d'un programme
- Results :Gestion des résultats d'un programme
- Explorer : Navigation entre les fichiers de données
  - Affichages par espace de stockage ou librairies
  - Détail des fichiers de données (nom, date de création, taille...)
  - Consultation des fichiers de données ou tables
  - Manipulation et création de fichiers de données
  - Gestion de l'espace de stockage (suppression des fichiers de données ou tables)
  - Création de librairies

Les fenêtres principales

- Footnotes : Permet de modifier les notes de bas de page des sorties SAS

- Options : Permet de modifier les options de la session en cours

- Keys : Permet de modifier les touches de raccourcis pour les commandes SAS

Les tables de données

- Le logiciel SAS permet de travailler sur des fichiers de données volumineux mais ces derniers doivent être mis sous forme de **tables ou tableaux SAS**

  - Stockées dans un **espace de stockage défini "les librairies"**,

  - Contenant des **individus** ou **observations** en ligne et une ou plusieurs **variables** en colonne

- Le nom d'une table doit être composé de 32 caractères maximum, il doit commencer par une lettre ou _ et peut contenir des chiffres.

L'espace de stockage

- SAS fonctionne avec des répertoires logiques qui sont différents des répertoires physiques.

- Pour SAS, il existe 2 ensembles de fichiers :

  - Des fichiers au format SAS:
    - manipulables en l'état à l'aide ou grâce à l'environnement SAS :
      - » Les tables de données
      - » Les vues qui sont des structures de données sans contenu physique
      - » Des catalogues qui sont des ensembles de stockage d'élémens SAS autres que des tables.Pour l'utilisateur courant, es éléments d'un catalogue seront principalement des formats, des écrans de saisie, des programmes et des graphiques (type = format, screen, source,grseg)
    - l'ensemble de ces fichiers constitue une **librairie**.

  - Les autres fichiers appelés « fichiers externes »
    - fichiers de type texte, Excels, dbase, html...

L'espace de stockage

- Gestion des finchiers SAS : **Les librairies**
  - **Elles permettent de stocker les tables SAS**. Chaque librairie est liée à un répertoire sur le disque dur ou sur le serveur utilisé.
  - Cela permet d'éviter la réécriture du chemin à chaque création de tables SAS en utilisant le nom de la librairie.

- Création d'une librairie :
  - Via l'instruction LIBNAME (fenêtre Editor):

LIBNAME *nom_lib* 'C:\Nom_repertoire';

  - Via la fenêtre Explorer par interface graphique.

*le nom de la librairie (nom_lib) ne doit pas exéder 8 caractères commençant par une lettre ou _ et pouvant contenir des chiffres

- Utilisation d'une librairie :

nom_lib.nom_table

L'espace de stockage

- Des options sont associés à l'instruction libname :
  - *List* fournira le chemin des répertoires physiques des librairies listés

LIBNAME *nom_lib* LIST;

  - *Clear* permettra la suppression des librairies listées

LIBNAME *nom_lib* CLEAR;

L'espace de stockage

- 2 types de stockage des fichiers SAS :
  - Stockage **permanent** :
    - Afin de créer des tables SAS permanentes, il est nécessaire de spécifier une « librairie », qui permette à SAS d'allouer des fichiers SAS
  - Stockage **temporaire** :
    - Lorsqu'aucune librairie n'est spécifiée lors de la création d'une table, SAS crée lui-même une librairie « WORK » dans laquelle il stocke des fichiers ou tables temporaires.
    - Attention cette librairie est éffacée en quittant SAS!

L'espace de stockage

- Gestion des fichiers externes :
  - Référence à un fichier externe :

FILENAME *nom_fic* 'C:\Nom_repertoire\fichier.txt';

  - Création d'un raccourci vers un répertoire de stockage des fichiers externes de sortie :

FILENAME *nom_rep* 'C:\Nom_repertoire\';

*le nom du fichier externe (nom_fic) et celui du raccourci (nom_rep) ne doit pas exéder 8 caractères commençant par une lettre ou _ et pouvant contenir des chiffres

Les variables

- Une table SAS contient des variables. Chaque variable est identifée par :
  - Son nom :
    - Il sera de préférence court car utilisé un grand nombre de fois mais explicite afin de faciliter l'identification d'une variable,
    - If faut éviter les caractères spéciaux ($+,...) et accentués,
    - Il ne faut pas utiliser les mots réservés par SAS
  - Son type : caractère ou numérique
  - Sa longueur:
    - Variable caractère :
      - ➥ Longueur par défaut : 8 caractères
      - ➥ Longueur mini :1 caractère, maxi : 32 767 caractères
    - Variable numérique (nombre d'octets sur lesquels une variable est stockée)
      - ➥ Longueur par défaut 8 octets
      - ➥ Longueur mini : 3 octets, maxi : 8 octets
  - Son format (format d'écriture)/ informat (format de lecture)
- Une variable, ensemble des valeurs concernant une même information, correspond à une colonne SAS

Types de traitements SAS

- 2 types d'étapes constituent un programme SAS :
  - L'étape DATA :
    - Elle permet de **construire et transformer** une ou plusieurs tables SAS
    - Elle commence par le mot clé **DATA** et se termine par l'instruction **RUN**
  - L'étape PROC :
    - Elle permet **d'exploiter** les tables SAS et de créer éventuellement des tables SAS de résultats
    - Elle commence par le mot clé **PROC** et se termine par l'instruction **RUN**
- Toutes les instructions se terminent par un **;**
- Des commentaires peuvent être insérés:
  - Soit il commence par ***** et se termine par **;**
  - Soit il commence par **/*** et se termine par ***/**

Options générales de SAS

- Des options générales sont associées à l'ouverture d'une session SAS :
  - **CENTER**/NOCENTER : centre l'impression des résultats des procédures
  - **DATE**/NODATE : imprime la date en haut de chaque page
  - **NUMBER**/NONUMBER : numérote les pages d'impression
  - **OBS=MAX** : indique la dernière observation à traiter
  - **FIRSTOBS=1** : indique la permière observtion à traiter
  - ...

  *En gras, les options par défaut.
- Il est possible de lister les options en utilisant la procédure suivante:

```
PROC OPTIONS; RUN;
```

- Les options sont modifiables à tout moment du programme par l'intruction (en général au début):

```
OPTIONS options;
```

L'instruction DM

- L'instruction DM permet de passer des commandes sous forme d'instruction permettant d'agir sur les différentes fenêtres SAS.

- Syntaxe :

DM *fenêtre* '' Instruction '';

  - Le nom de la fenêtre indique sur quelle fenêtre la commande va agir. Si aucun nom n'est mentionné, les commandes s'appliqueront à la fenêtre active.

- Exemple :

DM log “clear”;

  - Cette instruction efface le contenu de la log.

Sommaire

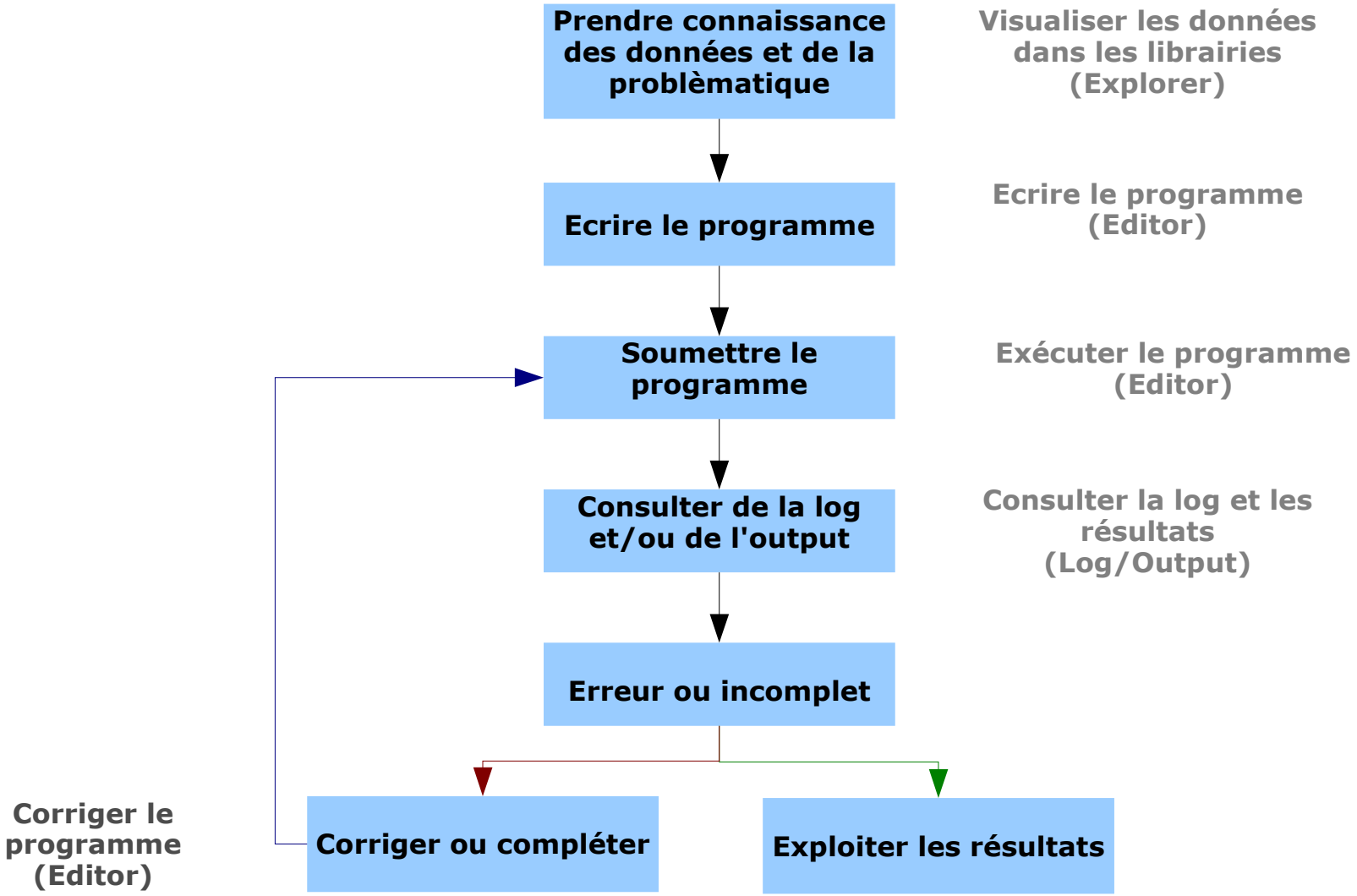
Partie 2

Manipuler et créer des tables SAS

1. Shéma de programmation
2. Prendre connaissance des données
3. L'étape DATA
4. Création d'une table SAS sur fichier extérieur
5. Les variables
6. Les procédures
7. Recodage des variables
8. Les instructions de gestion des variables
9. Manipulation des tables SAS

Schéma de programmation

- Comment bien programmer?



Prendre connaissance des données

- Avant de se lancer dans la programmation, il faut avant tout **comprendre la problèmatique** et **identifier** les besoins en terme de **données**.

- Après identification des données, il faut les localiser c'est à dire:
  - Identifier si elles sont disponibles dans les tables de travail?
  - Localiser dans quelles tables les données (ou variables) se trouvent?
  - Localiser dans quelles librairies les tables sont stockées?

Prendre connaissance des données

- Pour consulter le contenu d'une librairie, 2 solutions:
  - Via la fenêtre Explorer
  - Via une commande SAS :

```
PROC CONTENTS DATA=nom_lib._all_
DIRECTORY;
RUN;
```

- Pour consulter le contenu d'une table, 2 solutions:
  - Via la fenêtre Explorer
  - Via une commande SAS :

```
PROC CONTENTS DATA=nom_lib.table;
RUN;
```

L'étape DATA

- L'étape DATA permet de :
  - Créer des tables SAS suite à une recherche d'information
  - Créer des variables et recoder des données
  - Supprimer des variables
  - Concaténer des tables
  - Fusionner des tables
  - Mettre à jour des tables
  - Créer des fichiers externes

- Ces différentes actions se font à partir :
  - D'autres tables SAS
  - De fichiers externes
  - D'aucun fichier en entrée

L'étape DATA

- L'instruction DATA est la première instruction de l'étape Data, elle établit le nom de la table SAS en sortie.

- Syntaxe :

```
DATA   nom_lib.tableS    (liste options)
       nom_lib.tableS2   (liste options)
       ...
       nom_lib.tableSn   (liste options);
Liste instructions;
RUN;
```

- Cas particulier :
  - L'instruction DATA _Null_ ; ne crée pas de table en sortie.

L'étape DATA
L'instruction SET

- Cette instruction permet de créer une table à partir d'une autre table SAS.

- Syntaxe :

```
DATA   nom_lib.tableS    (liste options);
SET    nom_lib.tableE    (liste options);
Liste instructions;
RUN;
```

- Remarques:
  - Si le nom de table de sortie est identique à celui de la table d'entrée, cette dernière est écrasée.
  - L'instruction SET _LAST_; a la particularité de lire la dernière table SAS existante crée dans la session.

L'étape DATA
Les options de l'étape DATA

- Les options des tables SAS se mettent habituellement entre parenthèses après le nom d'une table, les plus courantes sont:
  - OBS=nn *(en entrée uniquement)*
    - détermine la fin du processus de l'étape à la nnième observation
  - FIRSTOBS=pp *(en entrée uniquement)*
    - détermine le début du processus de l'étape sur la ppième observation
  - END=varbooléenne *(en entrée uniquement)*
    - variable indicatrice qui permettra ensuite par une instruction conditionnelle appropriée d'éxécuter un ensemble d'instruction
  - KEEP= liste des variables
    - nom des variables à conserver dans la table en entrée ou en sortie
  - DROP= liste des variables
    - nom des variables à supprimer dans la table en entrée ou en sortie
  - WHERE=(Condition)
    - permet de filtrer les données en entrée ou en sortie
  - RENAME= (ancien nom=nouveau nom)
    - permet de renomer les variables en entrée ou en sortie

Création d'une table SAS sur fichier extérieur

- Comment importer un fichier extérieur?
  - 2 solutions:
    - Via la fenêtre EXPLORER
    - Via des instructions SAS dans la fenêtre EDITOR

- Syntaxe globale :

```
DATA    Fichier_SAS_créé_en_sortie;
INFILE  Fichier_en_entrée;
INPUT   Format_de_lecture;
RUN;
```

  - DATA permet de nommer la table SAS à créer
  - INFILE indique la référence du fichier à lire en entrée
  - INPUT précise comment lire les données

Création d'une table SAS sur fichier extérieur

- Instruction INFILE :
  - Elle permet d'identifier à l'intérieur d'un programme SAS le fichier extérieur (les données brutes) à transformer en table SAS.
  - Syntaxe : **INFILE *Nom_fichier_externe* (liste des options);**
    - La spécification du *nom_du_fichier* peut être:
      - Définit par l'instruction FILENAME
      - Le chemin accédant au fichier
    - Quelques options :
      - Firstobs = nnième observation : Numéro de la première observation du fichier en entrée à prendre en compte dans la table SAS créée.
      - Obs= nnième observation : numéro de la dernière observation du fichier en entrée à prendre en compte dans la table SAS créée.
      - DLM= 'séparateur' : Identification du séparateur des variables dans le fichier physique

        (DLM='09'x pour un séparateur tabulation, DLM=';' pour un sépérateur ; ,DLM=',' pour un séparateur ,)
      - DSD : Permet de lire un fichier avec des données manquantes et donc représentées par deux délimiteurs consécutifs.
      - MISSOVER : permet de lire des enregistrements avec des valeurs manquantes en fin d'enregistrement.

Création d'une table SAS sur fichier extérieur

- Instruction INPUT :
  - Elle permet de lire les données externes et de les placer dans le vecteur de travail et de décrire les variables à lire en précisant :
    - leur nom,
    - leur type,
    - leur longueur,
    - leur format de lecture (INFORMAT),
    - puis dans la table SAS créée, leur associer éventuellement un format d'édition (FORMAT) qui pourra être différent du format d'importation.
  - Il existe 3 modes INPUT :
    - Le mode liste : forme la plus simple de fichier d'entrée qui impose plusieurs restrictions sur les données (peu courant),
    - Le mode colonne : forme correspondant à un fichier avec des variables en colonne qui doivent toujours être dans les mêmes zones,
    - Le mode formatté : forme la plus courante,elle combine les caractéristiques du mode colonne et la lecture de données 'non-standards'.

Création d'une table SAS sur fichier extérieur

- Le mode liste :
  - Syntaxe :

INPUT var1 ($) var2 ($) ... varn ($);

    - Les noms des variables caractéres seront toujours suivis du symbole $.
    - Sur le fichier en entrée :
      - Les valeurs seront séparées par au moins un espace,
      - Si une valeur doit être manquante sur le fichier elle sera représentée par un .
      - Les champs alphanumériques ne doivent pas contenir d'espace ou de blancs
      - Ce format ne permet pas de lire le format de variable date.

Création d'une table SAS sur fichier extérieur

- Le mode colonne :
  - Syntaxe :

```
INPUT var1 ($) col_deb-col_fin (décimales) ...
      varn ($) col_deb-col_fin (décimales);
```

    - Les noms des variables caractéres seront toujours suivis du symbole $.
    - Sur le fichier en entrée :
      - Les valeurs des variables définies doivent toujours être dans les mêmes zones,
      - L'ordre des variables est indifférent,
      - Des sous zones de variables peuvent être lues.

Création d'une table SAS sur fichier extérieur

- Le mode formatté :
  - Syntaxe :

INPUT (pointeur) var1 ($) informat ...

(pointeur) varn ($) informat

(@ ou @@);

  - Pointeur : SAS permet de contrôler la colonne à partir de laquelle v se faire la lecture sur l'enregistrement en cours:
    - @n : v à la colonne n de la ligne courante
    - +n : avance de n colonnes vers la droite
    - #l : va à la ligne l à partir de la position actuelle
  - Informat (format de lecture) : il précise le type de la variable, sa longueur et le type de représentation utilisée (décimale, entier, date...)
  - En fin d'instruction :
    - @ : l'instruction INPUT déclenche la lecture de l'enregistrement suivant. Pour empêcher cette lecture, il suffit de mettre un @ en fin d'instruction et donc de reprendre la lecture du même enregistrement.
    - @@ : en fin d'instruction INPUT permet de rester sur le même enregistrement.

Création d'une table SAS sur fichier extérieur

- Procédure d'importation :
  - Syntaxe :

```
PROC IMPORT     DATAFILE= 'Chemin_fichier'
                OUT=table
                (DBMS=délimiteur) (replace);
                (GETNAMES=YES/NO);
              RUN;
```

  - DATAFILE : fichier en entrée (chemin du fichier ou nom logique)
  - OUT : table SAS créée
  - DBMS : délimiteur de données du fichier en entrée (XLS par défaut, première feuille)
    - ➥ CSV (fichier .CSV), TAB (tabulation, fichier .TXT), DLM (blanc, fichier .*)...
  - REPLACE : remplace la table en sortie si elle existe déjà
  - GETNAMES : indique si les libéllés des variables sont en première ligne

Création d'une table SAS sur fichier extérieur

- L'instruction CARDS permet de crée une table SAS à partir de la lecture en mode liste des enregistrements saisis u sein de l'étape DATA.
- Syntaxe :

```
DATA      Fichier_SAS_créé_en_sortie;
INPUT     var1 ($) var2 ($) … varn ($);
CARDS;
Liste_ des_valeurs_des_variables;
RUN;
```

  - DATA permet de nommer la table SAS à créer
  - INPUT précise comment lire les données (Voir mode liste)
  - CARDS permet de saisir à la suite les données à lire
- Les mêmes contraintes que le mode liste sont à appliquer.
- L'instruction INFILE CARDS MISSOVER; permet de lire des enregistrements avec des valeurs manquantes en fin d'enregistrement.

Les variables
Les formats d'écriture : FORMAT

- L'instruction FORMAT permet d'associer un format d'édition (écriture,affichage,impression) à une variable.
- Syntaxe:

FORMAT *nom_variable(s)* format.;

  - Il est possible d'affecter un même format à une liste de variable
  - Format. (voir liste des formats)

- Il est préférable de placer l'instruction FORMAT en début de l'étape DATA.

Les variables
Les formats de lecture : INFORMAT

- L'instruction INFORMAT permet d'utiliser un format de lecture ou de saisie pour une variable.
- Syntaxe:

INFORMAT *nom_variable(s)* informat.;

  - Il est possible d'affecter un même format à une liste de variable
  - Informat. (voir liste des informats)

- Il est préférable de placer l'instruction INFORMAT en début de l'étape DATA avant de faire référence aux variables concernées.

Les variables
Affecter un intitulé une variable : LABEL

- L'instruction LABEL permet de définir un intitulé clair pour une variable. Ce label sera permanent.

- Syntaxe :

LABEL *var1*=”label1” ... *varn*=”labeln”;

  - L'intitulé associé à une variable ne peut pas excéder 40 caractéres.
  - L'intitulé doit être entre apostrophes ou guillemets. Si une apostrophe apparaît dans ce dernier elle doit être doublée.

Les variables
Définir la longueur d'une variable LENGTH

- L'instruction LENGTH permet de définir la longueur (en octets) utile pour stocker des variables d'une table (par défaut 8 octets).
- Syntaxe :

LENGTH ***nom_variable(s)*** **($)longueur;**

  - Si la variable est de type caractère, la longueur est précédée de $.
  - Si la variable est de type numérique, il faut choisir judicieusement le nombre d'octets à associer:

Longueur de stockage	Valeur entière maximale stockable
3	8 192
4	2 097 152
5	536 870 912
6	137 438 953 472
7	35 184 372 088 832
8	9 007 199 254 740 990

Les variables
Définir les attributs des variables :ATTRIB

- L'instruction ATTRIB permet de définir pour une variable : le format de lecture (INFORMAT), le format d'écriture (FORMAT), un intitulé (LABEL) et une longueur de stockage (LENGTH).

- Syntaxe :

```
ATTRIB nom_variable
        INFORMAT    = informat.
        FORMAT      = format.
        LABEL       = "label"
        LENGTH      = ($) longueur;
```

  - Il est préférable de placer l'instruction ATTRIB avant toute autre instruction faisant référence aux variables concernées.

Les procédures
Généralités

- L'étape PROC est utilisée pour :
  - Effectuer des calculs statistiques,
  - Editer des états,
  - Sortir des graphiques,
  - Effectuer de la gestion de données,
  - Créer de nouvelles tables...

Les procédures
Généralités

- L'étape PROC possède deux sortes d'instructions:
  - L'instruction d'appel de la procédure:

PROC *nom_procedure* DATA=table_entrée *[options]*;

    - Si le nom de la table en entrée n'est pas précisé, la procédure prendra par défaut la dernière table créée.
  - Les instrutions de la procédure et les options des instructions:
    - Plusieurs instructions sont communes à différentes procédures:
      - BY : Traitement par sous-population
      - CLASS : Agrégation (variables qualitatives)
      - VAR : Sélection de variables (édition ou variables de calcul)
      - ID : Identification de l'observation
      - FORMAT : Associe une variable à un format d'édition
      - LABAL : Associe le nom d'une variable à un libellé en clair
      - WEIGHT et FREQ : Instructions de pondérations
  - Chaque procédure se finit par l'instruction RUN;

Les procédures
Exemple de procédures

- Procédures statistiques :
  - PROC FREQ (calcule de fréquence)
  - PROC MEANS, PROC SUMMARY (statistique descriptive de base)
- Procédures d'édition :
  - PROC TABULATE (construction de tableau)
  - PROC PRINT (impression de résultats dans l'output)
- Procédure graphique :
  - PROC GCHART
- Procédures utilitaires :
  - PROC DATASETS (liste, supprime et renomme les tables SAS)
  - PROC SORT (tri les tables)
  - PROC FORMAT (création de format)
  - PROC TRANSPOSE (transposition de table)
- Procédure de requête :
  - PROC SQL (utilisation du langage SQL)

http://wwwsas.stat.ucl.ac.be/sasdiscute/modules/basefr.html

Les procédures
Zoom sur la PROC FREQ

- Cette procédure permet d'étudier la distribution des modalités de variables ou de croisement de variables.

- Elle permet d'éditer :
  - Les modalités de la variable étudiée
  - Les fréquences d'apparition de chaque modalité
  - Le % global
  - Le % ligne
  - Le % colonne
  - Les fréquences cumulées
  - Les % cumulés

Les procédures
Zoom sur la PROC FREQ

- Syntaxe :

```
PROC FREQ DATA=table_entrée [options];
TABLE     var1 var2 var3 .. varn
          var1*var2
          var1*var2*var3 [/options];
```

  - Les options principales de l'instruction d'appel de la procédure sont:
    - ORDER=**INTERNAL** ou FREQ ou DATA ou FORMATTED (modifie l'ordre d'édition des modalités correspondant respectivement à un ordre croissant, décroissant, ordre d'apparition de la table ou ordre alphabétique des formats associés)
    - PAGE (permet d'éditer une variable ou un croisement de variables par page).
  - Les options de l'instruction TABLE sont :
    - Les options de suppression d'édition : NOFREQ (les effectifs), NOROW(% ligne), NOCOL (% colonne), NOPERCENT (% totaux), NOCUM (% cumulés), NOPRINT (pas d'édition dans l'output)
    - Les autres options : LIST MISSING OUT=table_sortie

Les procédures
Zoom sur la PROC MEANS

- La procédure MEANS permet de produire un certain nombre de calculs statistiques sur l'ensemble d'une table ou sur un groupe d'observations.

- La procédure MEANS sans option et sans instruction de procédure édite pour chaque variable numérique de la table:
  - N = nombre d'observation où la valeur est non manquante
  - MEAN = moyenne
  - STD = écart-type
  - MIN = valeur minimum
  - MAX = valeur maximum

Les procédures
Zoom sur la PROC MEANS

- Syntaxe :

```
1   PROC MEANS DATA=table_entrée [options];
2   CLASS      varc1 varc2 varc3 .. varcn;
3   VAR        varn1 varn2  varn3 … varnn;
4   WEIGHT     varp;
        ou
    FREQ       varp;
5   OUTPUT     OUT=table_sortie (options)
               stat1(varn1 varn2…)=varstat1 varstat2 …
               stat2(varn1)=varstat21….;
    RUN;
```

Les procédures
Zoom sur la PROC MEANS

- Descriptif des instruction:
  - 1 : Instruction d'appel de la procédure, les options sont les suivantes:
    - MISSING : prend en compte les valeurs manquantes dans les calculs
    - NOPRINT: n'affiche pas les résultats dans la fenêtre OUTPUT
    - NWAY : ne garde que les résultats statistiques du croisement le plus fin
  - 2 : Instruction servant à définir des sous-populations pour les quelles on veut des calculs statistiques (variables qualitatives)
  - 3 : Instruction servant à déclarer les variables quantitatives sur lesquelles seront calculées les statistiques.
  - 4 : WEIGTH pondére les résultats des calculs statistiques, FREQ pondére les résultats des calculs statistiques ainsi que la population de la classe.
  - 5 : Instruction permettant de conserver tous les calculs statistiques dans une table.

Les procédures
Zoom sur le PROC SORT

- Cette procédure permet de classer les observations dans l'ordre d'une ou plusieurs variables de la table.
- Syntaxe :

```
PROC SORT DATA=table_entrée [OUT= table_sortie  options];
    BY var1 var2 … varn ;
    RUN;
```

- Les options principales sont :
  - NODUPKEYS = élimine les observations ayant le même clé de tri
  - NODUPREC = élimine les doublons complets
  - FORCE = supprime l'index en sortie dans un tri
- Par défaut, le tri est croissant. Pour le rendre décroissant, il faudra utiliser l'option DESCENDING avant le nom de la variable à trier ainsi.

Les procédures
Zoom sur le PROC SORT

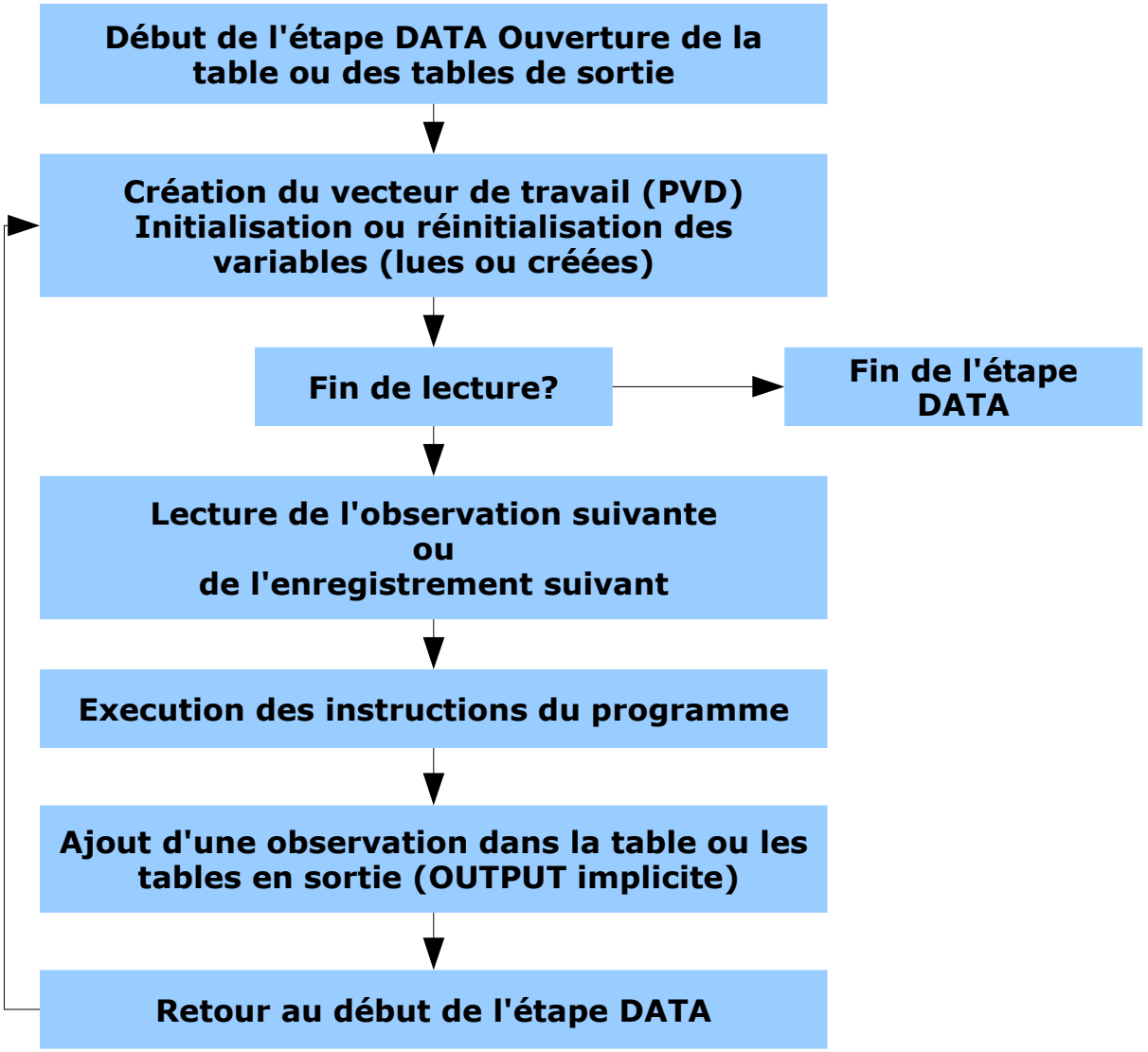
- Cette procédure permet de classer les observations dans l'ordre d'une ou plusieurs variables de la table.
- Syntaxe :

```
PROC SORT DATA=table_entrée [OUT= table_sortie  options];
    BY var1 var2 … varn ;
    RUN;
```

- Les options principales sont :
  - NODUPKEYS = élimine les observations ayant le même clé de tri
  - NODUPREC = élimine les doublons complets
  - FORCE = supprime l'index en sortie dans un tri
- Par défaut, le tri est croissant. Pour le rendre décroissant, il faudra utiliser l'option DESCENDING aant le nom de la variable à trier ainsi.

Recodage des variables
Fonctionnement de l'étape DATA

- Schéma de l'étape DATA:



Recodage des variables
Généralité

- Une nouvelle variable est créée dans une table par une instruction d'affectation dans l'étape DATA.

- Une nouvelle variable est créée à partir :
  - D'une variable SAS existant déjà,
  - D'une constante (numérique, caractère ou date),
  - D'une expression SAS (avec ou sans fonction).

  **Liste des opérateurs et fontions*

- Remarque :
  - La variable de totalisation permet d'effectuer des cumuls, de gérer des compteurs. Elles ne réinitialisent pas lors de l'éxécution de l'étape DATA et cumulent donc les valeurs.
  - Plusieurs syntaxes :
    - VAR1+VAR2; *obligatoirement numérique*
    - VAR1+constante numérique;
    - VAR1+expression arithmétique;

Les Instructions de gestion des variables

- Certaines options de l'étape DATA sont aussi des instructions, elles se placent après l'instruction DATA:
  - KEEP var1 var2 ... varn;
  - DROP var1 var2 ... varn;
  - RENAME ancien nom=nouveau nom;
- Les instructions filtres sont :
  - IF condition;
    - L'instruction IF teste une condition. Si la condition n'est pas remplie les instructions suivantes de l'étape DATA ne sont pas effectués.
  - WHERE condition;
    - L'instruction WHERE teste une condition. Si la condition n'est ps remplie, l'observation de la table en entrée n'est pas chargée dans le vecteur de travail.
    - Tous les opérateurs SAS sont valables pour le WHERE. Des opérateurs supplémentaires sont valables:
      - ➥ CONTAINS : opérateur sur les chaînes de caractères,
      - ➥ LIKE avec % remplaçant plusieurs caractères ou _ remplaçant 1 caractère.
      - ➥ Variable BETWEEN val1 AND val2;
      - ➥ Variable IS NULL;

Les Instructions de gestion des variables

- Le traitement conditionnel se fait avec l'instruction IF :
  - IF condition1 THEN instruction1;
    ELSE IF condition2 THEN instruction2;
    ELSE instruction3;
  - IF condition THEN DO; instructionc11;instructionc12;...;END;
    ELSE DO; instructionc21;instructionc22;...;END;

- L'instruction DELETE provoque le retour au début de l'étape DATA sans écriture de l'observation courante:
  - IF condition THEN DELETE;

- L'instruction OUTPUT est implicite en fin de boucle de l'étape DATA, cependant il est possible de gérer l'écriture des observations dans une ou plusieurs tables en sortie:
  - IF condition THEN OUTPUT tables1;
    ELSE OUTPUT tabes2;

Les Instructions de gestion des variables

- Le traitement itératif se fait avec l'instruction DO :
  - Cette instruction se finit obligatoirement par un END;
  - Boucle simple:

```
DO i=ind1 TO ind2 BY pas;
    Instruction1;
    ...
    Instruction n;
END;
```

  - Boucle *tant que* dont la condition est évaluée avant l'éxécution des instructions de la boucle:

```
DO WHILE (condition);
    Instruction1;
    ...
    Instruction n;
END;
```

Les Instructions de gestion des variables

- Le traitement itératif se fait avec l'instruction DO :
  - Boucle *jusqu'à ce que* dont la condition est évaluée aprés l'éxécution des instructions de la boucle:

```
DO UNTIL (condition);
    Instruction1;
    ...
    Instruction n;
END;
```

- Les instructions du DO UNTIL sont exécutées au moins une fois mais pas celles du bloc DO WHILE.

Manipulation de tables SAS
Instruction SET

- Concaténation de table :
  - L'instruction SET permet de transférer les données d'une ou plusieurs tables SAS dans le vecteur de travail.
  - Elle permet aussi d'intercaler les observations de 2 ou plusieurs tables SAS, à condition qu'elles aient la même structure et qu'elles soient triées. Il faut recourir à l'instruction complémentaire BY.
  - Syntaxe:

```
DATA    nom_lib.tableS (liste options);
SET     nom_lib.tableE1 (liste options)
        nom_lib.tableE2 (liste options) ...
        nom_lib.tableEN (liste options);
BY var_cle;
Liste instructions;
RUN;
```

Manipulation des tables SAS
Instruction SET

- L'option IN:
  - L'instruction SET, utilisée avec l'option IN (assimiliée à une variable logique temporaire, valable uniquement durant l'étape DATA) sert à indiquer la provenance de l'observation. Quand l'observation est présente dans la table elle prend la valeur 1, sinon elle est à 0.
  - Syntaxe:

```
SET      nom_lib.tableE (IN=nom_var1 liste options);
```

- L'option END:
  - L'instruction SET, utilisée avec l'option END (assimiliée à une variable logique temporaire, valable uniquement durant l'étape DATA) indique la dernière observation lue. La variable attribuée à END est initialisée à 0 et prend la valeur 1 à la dernière observation de la table.
  - Syntaxe:

```
SET      nom_lib.tableE (END=nom_var1 liste options);
```

Manipulation des tables SAS
Instruction SET

- L'utilisation des instructions BY, RETAIN, FIRST et LAST:
  - Ces instructions permettent de calculer des agrégats.
  - Exemple de syntaxe :

```
DATA     nom_lib.tableS     (liste options);
SET nom_lib.tableE1 (liste options)
            nom_lib.tableEN (liste options);
BY var_cle;
RETAIN var_ind1 var_ind2 … var_indN;
IF FIRST.var_cle THEN DO; var_ind1=0; var_ind2=0; …; var_indN=0;END;
Liste instructions;
IF LAST.var_cle THEN DO; instructions; END;
RUN;
```

Manipulation des tables SAS
Instruction SET

- L'utilisation des instructions BY, RETAIN, FIRST et LAST:
  - L' instruction BY permet de trier au préalable selon la ou les variables spécifiées. Elle génére 2 variables temporaires dans le PVD : FIRST.var_cle et LAST.var_cle où var_cle est la ou les variables spécifiées par l'option BY.
  - FIRST.var_cle vaut 1 chaque fois que var_cle change de valeur et vaut 0 pour toutes les autres observations.
  - LAST.var_cle vaut 1 pour la dernière observation et 0 pour toutes les autres.
  - RETAIN permet de retenir le contenu des variables calculées et donc d'éviter leur réinitialisation.

Manipulation des tables SAS
Instruction MERGE et BY

- L'instruction MERGE permet de fusionner des observations de 2 ou n tables SAS en une seule table SAS.
- L'instrucion BY permet de réunir dans la même observation d'une nouvelle table, les observations de tables SAS ayant la même valeur de clé.
- Les tables en entrée doivent être triées sur le même critére de fusion.
- Syntaxe:

```
DATA   nom_lib.tableS (liste options);
MERGE  nom_lib.tableE1 (liste options)
       nom_lib.tableE2 (liste options) ...
       nom_lib.tableEN (liste options);
BY var_cle;
Liste instructions;
RUN;
```

Manipulation des tables SAS
Instruction MERGE et BY

- L'option IN:
  - L'option IN indique à SAS si la modalité courante de la clé de fusion existe dans la table en entrée.
  - Elle permet d'utiliser une variable logique qui prend la valeur 1 si la modalité est dans la table ou 0 si elle n'y est pas.
  - L'utilisation de ces variables temporaires peuvent être utilisé pour filtrer les données en sortie.
  - Syntaxe :

```
DATA    nom_lib.tableS (liste options);
MERGE   nom_lib.tableE1 (liste options IN=varint1)
        nom_lib.tableE2 (liste options IN=varint2) ...
        nom_lib.tableEN (liste options IN=varintn);
BY var_cle;
Liste instructions;
RUN;
```

Création de format d'édition
PROC FORMAT

- Cette procédure permet de créer des formats d'édition sur a base de valeurs contenues dans une variable.
- Les formats se définissent dans la procédure FORMAT.

- Syntaxe:

```
PROC FORMAT LIBRARY=nom;
    VALUE  $nom du format
        val1              =   "Groupe1"
        val2,val3,val4=   "Groupe2"
        val5-val9         =   "Groupe3" ...;
    RUN;
```

- Remarque:
  - Une procédure FORMAT peut contenir plusieurs instructions VALUE.

Création de format d'édition
PROC FORMAT

- Il existe 3 valeurs particulières :
  - LOW : désigne la valeur la plus petite (hormis la valeur manquante)
  - HIGH : désigne la valeur la plus grande
  - OTHER : désigne les valeurs non précédemment décrites.

- La procédure FORMAT n'édite rien, elle stockeles formats dans le catalogue de formats indiqué par LIBRARY=nom. Si aucune librairie n'est précisée alors le format créé est stocké dans le catalogue work.formats.

Création de format d'édition
Création d'une table à partir d'une table SAS

- 1 ére étape : Création de la table SAS de référence
  - La procédure FORMAT n'a besoin que de 4 variables pour pouvoir préparer la création d'un format. Il suffira donc de créer une table SAS contenant ces variables:
    - FMTNAME : variable contenant le nom du format
    - TYPE : variable de 1 caractère indiquant le type de format (C = caractère, N=numérique)
    - LABEL : variable contenant la variable formattée
    - START : variable contenant la valeur que l'on veut formater (ou START et END varaibles contenant les bornes inférieures et supérieures incluses)
- 2 ème étape : Création du format
  - Il reste à éxécuter la procédure FORMAT sur cette table SAS pour charger le format.
  - Syntaxe: CNTLIN pour la création, CNTLOUT pour la lecture

```
PROC FORMAT CNTLIN=nom de table;RUN;
```

```
PROC FORMAT CNTLOUT=nom de table;
SELECT format;RUN;
```