



Initiation au logiciel SAS



Manel BEN ALI
manel.benali.essait@gmail.com

Objectifs

- ▶ Proposer une entrée en matière à l'utilisation du logiciel SAS
- ▶ Illustrer divers éléments de Statistique élémentaire

SAS étant un logiciel complet largement adopté dans l'industrie, il est à noter qu'une bonne maîtrise du logiciel est un pré-requis récurrent dans les offres de stages et d'emplois destinées aux étudiants de filière Statistique



Plan du cours

Introduction

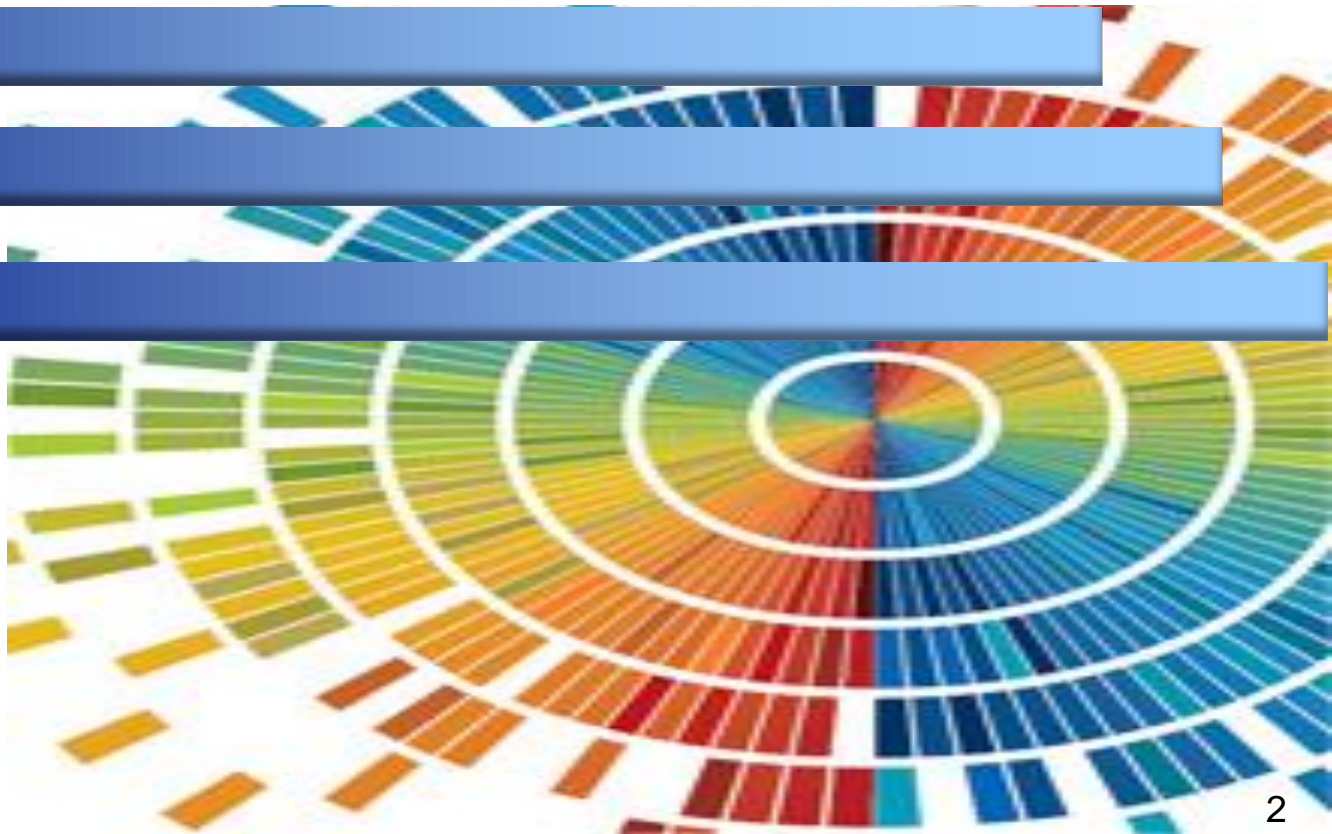
Les Etapes DATA

Les Procédures

SQL

SAS Graph

SAS Macro



Introduction

Le système **SAS®**, (*Statistical Analysis System*) est un **logiciel de statistique polyvalent** constitué par un ensemble de modules pour la gestion et le traitement statistique des données provenant de différents systèmes opérationnels afin de fournir des informations pertinentes. SAS® a été conçu en 1976 par la société SAS® Institute Inc. basée à Cary, en Caroline du Nord.

Le système SAS®, c'est :

- Un système de **gestion de données**

Les informations sont stockées dans une structure propre au système SAS®, la table SAS. La manipulation des données s'appuie sur un langage de quatrième génération, le langage SAS.

- Un système de **traitement de données**

La plupart des traitements envisageables sont disponibles: statistiques, financiers, industriels, mathématiques, etc.

- Un accès à **tous types de données**

SAS accède de façon transparente à la plupart des SGBD du marché (DB2, Oracle, Ingres, RDB, SQL/DS, etc.).



Raisonner en « SAS »



Qu'est-ce qu'un programme?

Algorithme, recette, jeu d'instructions

► ***SAS est comme la programmation dans n'importe quel langage:***

Instructions étape par étape

Possibilité de créer vos propres routines pour traiter les données

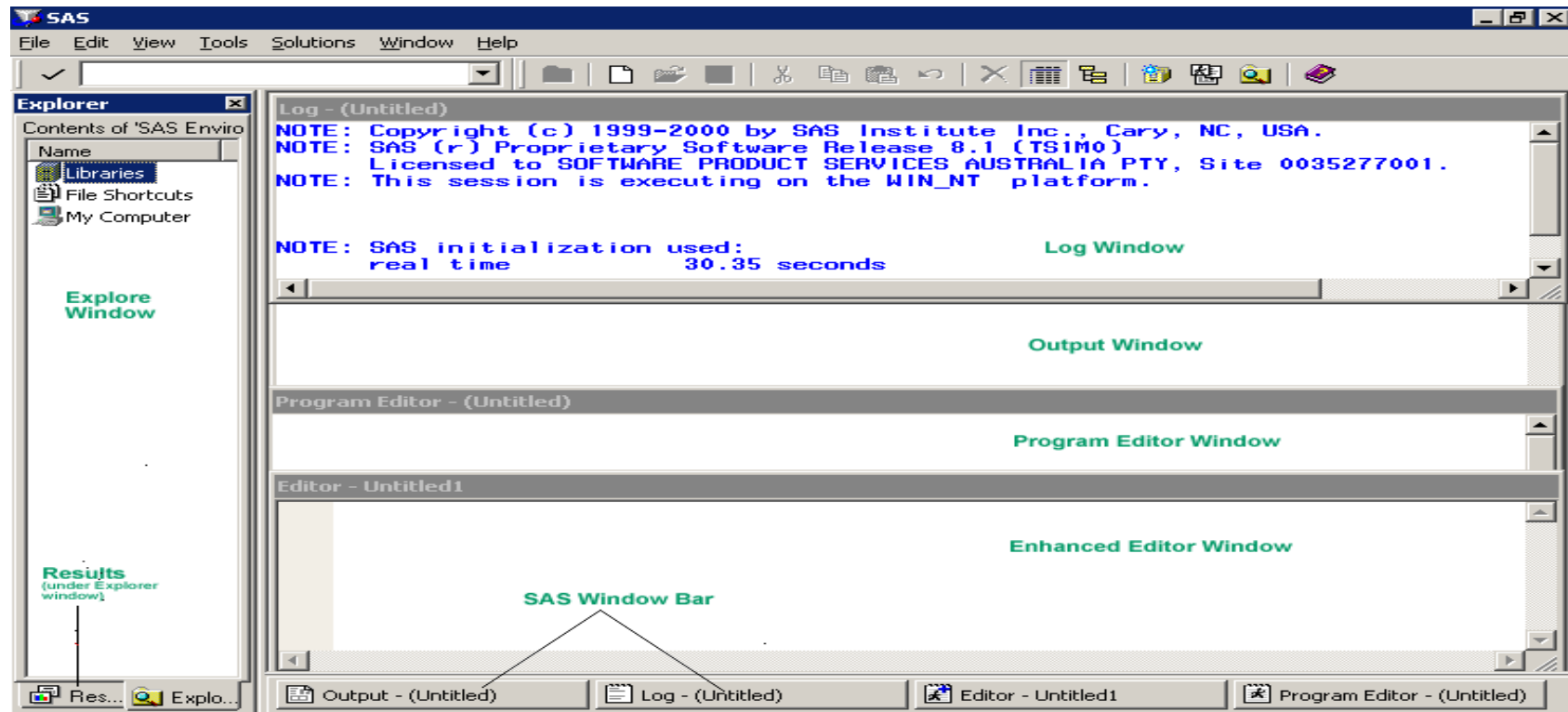
La plupart des instructions sont mises en place de manière logique

► ***SAS n'est pas comme les autres langage:***

Certaines syntaxes sont propres à SAS

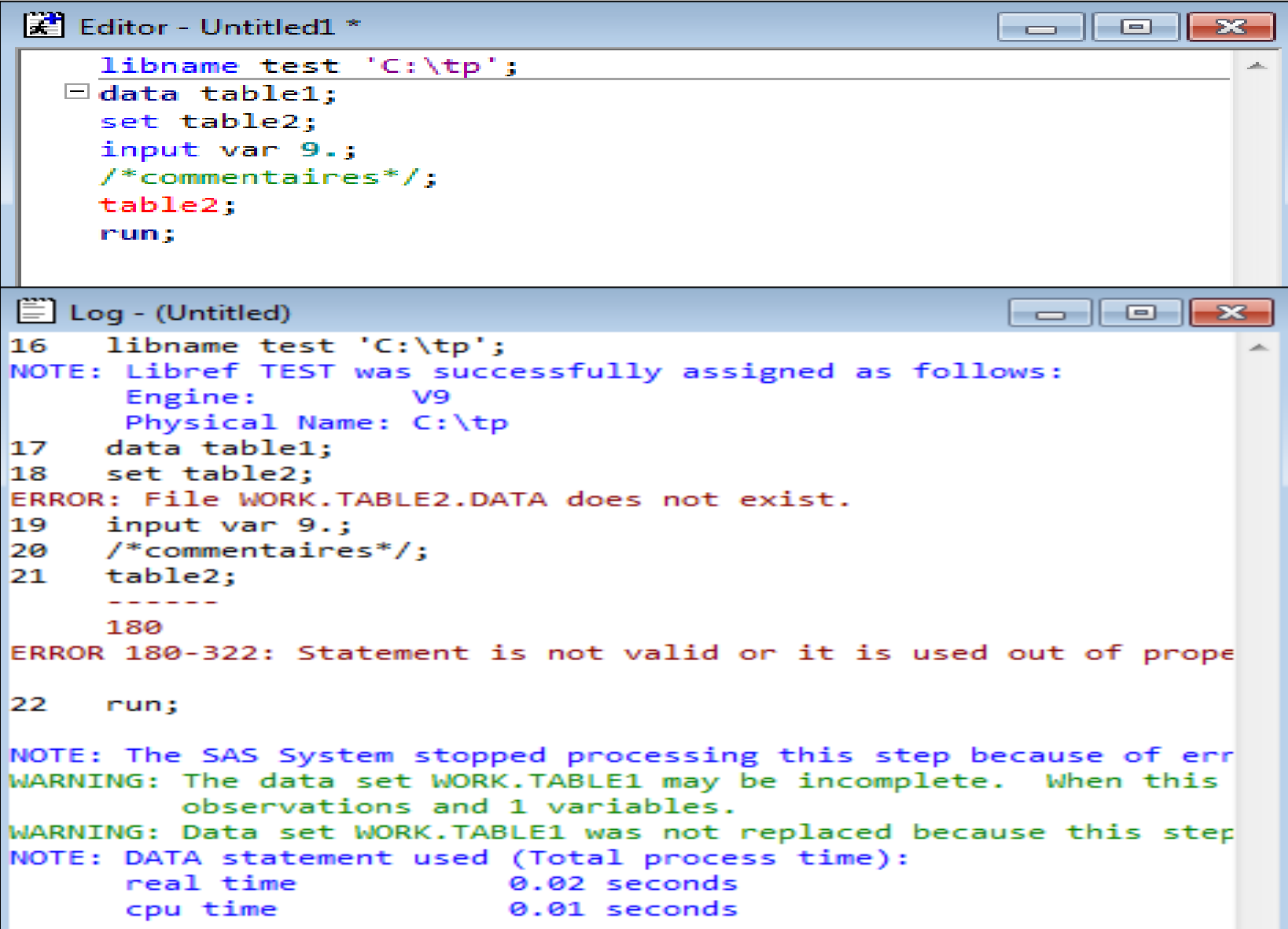
Ecrit spécialement pour les statistiques de sorte qu'il ne soit pas tout-usage .

Environnement SAS : Fenêtres



- Log /Journal : Vérifier l'exécution
- Editor/Editeur : Écrire le code
- Output/Sortie : Affiche les résultats
- Explorer/Explorateur :Afficher l'environnement SAS : Bibliothèques

Environnement SAS : Couleurs



The screenshot displays two windows from the SAS environment. The top window, titled 'Editor - Untitled1 *', contains SAS code with syntax highlighting: `libname test 'C:\tp';` (blue), `data table1;` (blue), `set table2;` (blue), `input var 9.;` (blue), `/*commentaires*/;` (green), `table2;` (red), and `run;` (blue). The bottom window, titled 'Log - (Untitled)', shows the execution log with the same code lines numbered 16 to 22. It includes a note about the library assignment, an error message 'ERROR: File WORK.TABLE2.DATA does not exist.' on line 21, a subsequent error 'ERROR 180-322: Statement is not valid or it is used out of proper syntax' on line 22, and several warnings and notes regarding the data set and processing time.

```
libname test 'C:\tp';
data table1;
set table2;
input var 9.;
```

```
/*commentaires*/;
table2;
run;
```

```
16  libname test 'C:\tp';
NOTE: Libref TEST was successfully assigned as follows:
      Engine:          V9
      Physical Name: C:\tp
17  data table1;
18  set table2;
ERROR: File WORK.TABLE2.DATA does not exist.
19  input var 9.;
```

```
20  /*commentaires*/;
21  table2;
-----
180
ERROR 180-322: Statement is not valid or it is used out of proper syntax.
22  run;
```

```
NOTE: The SAS System stopped processing this step because of error.
WARNING: The data set WORK.TABLE1 may be incomplete.  When this step was
        executed, it had 0 observations and 1 variables.
WARNING: Data set WORK.TABLE1 was not replaced because this step did not
        change it.
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.01 seconds
```

Librairie SAS

- ❑ Raccourci pointant vers un répertoire.
- ❑ Lors de l'invocation de SAS, on a automatiquement accès à des bibliothèques de données SAS (temporaires et permanentes)
- ❑ Au cours d'une même session toutes vos données sont par défaut sauvegardée dans une bibliothèque **temporaire** (WORK) qui est vidée à chaque fermeture du logiciel SAS.
- ❑ Une bibliothèque permanente permet de sauvegarder vos données et/ou vos macros pour pouvoir les réutiliser facilement.
- ❑ Une table **permanente** est stockée dans une librairie. La table temporaire est détruite à la fin de la session SAS en cours.

Syntaxe

La librairie est définie par une instruction LIBNAME et assurant la correspondance entre le nom de librairie SAS et le nom physique (répertoire) de la librairie :

```
LIBNAME NomLIB 'chemin' ;
```

Exemple : On stocke de façon permanente la table temporaire Russet dans la librairie pays.

```
DATA pays.Russet ;
```

```
SET Russet ;
```

```
RUN ;
```

*Règles pour nommer une librairie SAS:

*Le nom doit avoir 8 caractères au maximum.

*Le nom doit commencer avec une lettre ou _

La table SAS

	ID	Surname	GivenNam	Street	City	State	Country	PostalCo	Phone
1	101	Devlin	Michaels	114 Pioneer Avenue	Kingston	NJ	USA	07070	2015558966
2	102	Reiser	Beth	88 Whippany Road	Rockwood	NY	USA	10154	2125558725
3	103	Niedringhaus	Erin	190 Windsor Street	Tara	PA	USA	19301	2155556518
4	104	Mason	Meghan	5520 Dundas Street East	Cheslea	TN	USA	37919	6155555468

- Nom** obligatoire :maximale 32 caractères
- Type** obligatoire :numérique (type par défaut) ou caractère
- Libellé** (label) facultatif: maximale 256 caractères
- Longueur** (length) obligatoire
- Format** facultatif : définit le format d'écriture de données
- Informat** facultatif : définit le format de lecture de données

Principe du langage SAS

Un programme SAS est constitué d'une succession d'étapes que l'utilisateur soumet au moteur SAS pour exécution.

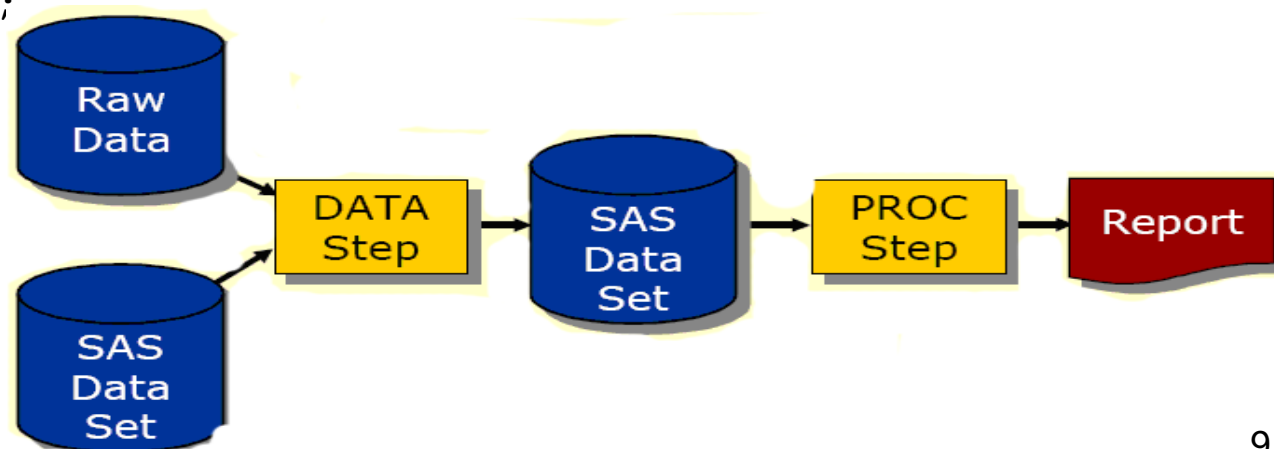
On distingue deux types d'étapes :

L'étape **DATA** : manipulation des données / Création et modification de table SAS

```
DATA output_DATA;  
  SET input_DATA;  
  /* [manipulation données] */  
RUN;
```

L'étape **Proc** : Analyses statistiques de bases de données, des graphiques / Reporting

```
PROC PRINT DATA = output_DATA ;  
  * [code spécifique] ;  
RUN;
```



L'étape DATA

- ☑ Import des données
- ☑ Saisie des données sous SAS
- ☑ Création d'un fichier texte à partir d'une table SAS
- ☑ Manipulation des données
- ☑ Chargement et fusion de tables SAS



L'étape DATA

Origine des données



Fichier

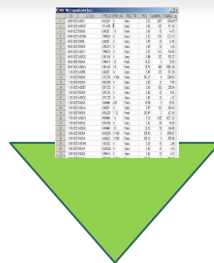


Table SAS



Papiers

Étape DATA

**Table
SAS**

Origine des données : *Papiers*

Saisie manuelle des données

Syntaxe

```
DATA    nom_table ;  
  Input  variables;  
  
  Cards;  
  ...Données entrées "à la main" ...  
Run;
```

Exemple

```
DATA Moyenne;  
  input nom $ moy1 moy2;  
  cards;  
  Pierre 10 12  
  Paul 11 13  
  Jacques 8 14  
RUN;
```

→ La table SAS à créer nommée `nom_table`

→ **Input** : Déclaration des variables (position de la variable, son type - caractère (suivi de \$) ou numérique - , nom de la variable)

→ **Cards** : Mot clé indiquant que l'on va saisir des données

Origine des données : *Table SAS*

Lecture d'un fichier en format SAS

Syntaxe:

```
DATA  nom_table_sortie;  /*      (nouvelle table )*/  
  SET    nom_table_source;  /* (Tableau SAS en entrée ) */  
  /* [manipulation DATA] */  
RUN;
```

- ❑ Une observation du tableau d'entrée est lue à chaque exécution de l'instruction SET.
- ❑ S'il y a plusieurs tableaux, on lit d'abord les observations du premier tableau avant de lire le suivant.

Origine des données : *Fichier*

Import de données

Syntaxe

```
DATA nom_table; /* (nouvelle table )*/
  INFILE 'chemin du fichier en entrée' <options> ; /* indique la référence
    du fichier à lire en entrée*/
  INPUT var1 var2;
RUN;
```

Exemple :Création d'une table SAS à partir d'un fichier texte

```
DATA moyenne;
  Infile 'C:\tp1\moyenne.txt'    firstobs=2 obs=2;
  Input nom $ moy;
Run;
```

FIRSTOBS=numéro_d'observation

Numéro de la première observation du fichier en entrée à prendre en compte dans la table SAS créée.

OBS=numéro_d'observation

Numéro de la dernière observation du fichier en entrée à prendre en compte dans la table SAS créée.

MISSOVER

En cas d'enregistrements de longueur variable, si toutes les variables de l'ordre INPUT n'ont pas été lues, les variables non renseignées sont mises à valeur manquante.

Les contrôleurs de pointeur

- ❑ @n : déplace le pointeur à la colonne n spécifiée.
- ❑ +n : déplace le pointeur à n colonnes vers la droite.
- ❑ #n : déplace le pointeur à la ligne n spécifiée.
- ❑ _N_ : n^{ème} étapes du block DATA.

- ❑ A la fin de l'énoncé **Input** :
 - ❑ @ : rester sur la ligne de données courante pour le prochain énoncé INPUT .
 - ❑ @@ : rester sur la ligne courante pour des exécutions futures du bloc DATA.

Traitement de données

Création et manipulation de variables

- Le signe **(=)** : Calcul de variables

- L'instruction **length**

```
Length var <$ si caractère > n;      n:longueur de la variable
```

- L'instruction **format**

```
format var <$>n;
```

- L'instruction **Keep** : garder une variable des données

```
DATA nom_tab( keep =var1 var2);
```

- L'instruction **Drop** : supprimer une variable des données

```
DATA nom_tab( drop =var1 var2);
```

- L'instruction **Rename** :renommer une variable

```
DATA nom_tab(rename =(ancien_nom=nouveau_nom)); ou bien
```

```
Rename ancien_nom=nouveau_nom ;
```

- L'instruction **Where**

```
DATA nom_tab(where =(expression)); ou bien where expression ;
```

Traitement de données

Manipulation des observations

L'instruction **delete**

- suppression des valeurs (manquantes , aberrantes..)

```
if condition then delete;
```

n_client	var1	var2
1	1	0
1	0	0
1	0	1
2	1	1
3	1	0
3	0	1



n_client	var1	var2
1	0	0
1	0	1
3	0	1

L'instruction **output**

- affiche la table nomtab sous la condition « condition1 »

```
if condition1 then output nomtab;
```

Traitement de données

Exemple : *Manipulation des données au cours d'une étape DATA*

Dans un fichier pays.russet, on sélectionne uniquement les démocraties

Obs	Pays	Demo
1	Australie	1
2	Russie	0
3	Autriche	2
4	Belgique	1
5	Egypte	0

```
DATA demo;  
  SET pays.russet;  
  IF demo=1 or demo=2;  
RUN;
```

De manière équivalente, on peut utiliser l'instruction **WHERE** :

```
DATA demo;  
  SET pays.russet;  
  WHERE demo=1 or demo=2;  
RUN;
```

Traitement de données

L'instruction **RETAIN**

permet de « protéger » la valeur d'une variable dans le vecteur de travail. On peut donc accéder, d'une observation à l'autre, à la valeur de la variable telle qu'elle était pour l'observation précédente. RETAIN s'applique surtout à de nouvelles variables .

```
retain var valeur-initiale;
```

```
DATA Comptes_an;
```

```
set Comptes_an;
```

```
retain Cumul_Compte 0;
```

```
Cumul_Compte = Cumul_Compte + N_compte_Ouvert;
```

```
Run;
```

	Année	N_compte_Ouvert
1	2000	100
2	2001	150
3	2002	200
4	2003	70
5	2004	160
6	2005	110
7	2006	140
8	2007	170
9	2008	52
10	2009	135
11	2010	190



	Année	N_compte_Ouvert	Cumul_Compte
1	2000	100	100
2	2001	150	250
3	2002	200	450
4	2003	70	520
5	2004	160	680
6	2005	110	790
7	2006	140	930
8	2007	170	1100
9	2008	52	1152
10	2009	135	1287
11	2010	190	1477

Traitement de données

L'instruction **RETAIN**

Cas 1 : incrémentation

Dans ce cas, on souhaite incrémenter une variable (NUMERO_ELEVE) à chaque observation.

Sans RETAIN, on perdrait la valeur précédente de NUMERO_ELEVE en changeant d'observation. Le résultat serait donc, sans RETAIN : à la 1ère observation, NUMERO_ELEVE vaudrait 1, et à toutes les autres observations, cette variable serait manquante.

(car valeur manquante + 1 \longrightarrow valeur manquante)

```
DATA work.class ;  
  SET sashelp.class ;  
  RETAIN numero_eleve ;  
  IF _N_ = 1 THEN numero_eleve = 0 ;  
  numero_eleve = numero_eleve + 1 ;  
RUN ;
```

Traitement de données

L'instruction **RETAIN**

Cas 2 :Stockage de la valeur d'une variable

Pour chaque client, on a l'historique de ses achats. On veut calculer, pour chaque client, l'ancienneté de ses achats par rapport au tout premier qu'il a effectué.

```
PROC SORT DATA = work.ventes ;
  BY numClient dateAchat ;
RUN ;
DATA work.ventes ;
  SET work.ventes ;
  BY numClient ;
  RETAIN premierAchat ;
  FORMAT premierAchat DDMMYY10. ;
  IF FIRST.numClient
  THEN premierAchat = dateAchat ;
  delai = dateAchat - premierAchat ;
RUN ;
```

numClient	dateAchat	premierAchat	delai
1	14/06/1999	14/06/1999	0
1	25/07/1999	14/06/1999	41
1	25/10/1999	14/06/1999	133
1	06/07/2001	14/06/1999	753
2	01/06/2000	01/06/2000	0
2	19/12/2000	01/06/2000	201

Ici, **RETAIN** permet de conserver la mémoire de la première date d'achat par client (repéré par un booléen **FIRST** disponible grâce au SET ... BY) en la stockant dans une nouvelle variable

Les structures conditionnelles

IF.....THEN.....ELSE

Syntaxe

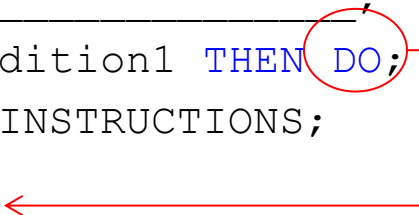
```
DATA _____;  
  SET _____;  
  IF condition1 THEN statement1;  
  ELSE statement2;  
  IF condition2 THEN statement3;  
  ELSE IF condition3 THEN statement4;  
  ELSE IF condition4 THEN statement5;  
  ELSE statement6;  
RUN;
```

Les structures conditionnelles

IFTHEN DO :

Syntaxe

```
DATA _____;  
  SET _____;  
  IF condition1 THEN DO;  
  LISTE INSTRUCTIONS;  
  END;  
RUN;
```



Chaque **DO** demande **END** pour fermer la boucle

Export de données

Création d'un fichier texte à partir d'une table SAS

Assigner un fichier de sortie par une instruction **FILENAME**

Créer une étape DATA _NULL_ (l'objectif n'est pas de construire une table SAS, mais de recopier les éléments d'une table SAS dans un fichier externe)

Formater les variables à l'aide de l'instruction PUT

L'instruction PUT s'utilise avec la même logique que l'instruction INPUT. Elle permet d'écrire les données en utilisant éventuellement un format d'écriture. La table SAS pays.russet est recopiée dans un fichier texte.

```
FILENAME russet 'c:\Odile Wolber\Cours_SAS\Russet.txt';  
DATA _NULL_; /*Pas de table SAS crée*/  
  SET pays.russet;  
  FILE russet;  
  put pays $ Gini Farm Demo;  
RUN;
```

ou de manière équivalente :

```
DATA _NULL_;  
  SET pays.russet;  
  FILE 'c:\Odile Wolber\Cours_SAS\Russet.txt';  
  put pays $ Gini Farm Demo;  
RUN;
```

Export de données

Création d'un fichier Excel /Texte à partir d'un fichier SAS avec Proc export

Export en fichier texte

```
PROC EXPORT DATA = tableSAS  
  OUTFILE = "chemin et nom du fichier créé"  
  DBMS = DLM REPLACE ;  
  DELIMITER = " " ; /* séparateur espace */  
RUN ;
```

Export en classeur Excel

```
PROC EXPORT DATA = tableSAS  
  OUTFILE = "chemin et nom du fichier créé"  
  DBMS = EXCEL REPLACE ;  
  SHEET = "nomFeuille" ; /* nom de la feuille Excel créée */  
RUN ;
```

Dates sous SAS

Comment sont stockées les dates SAS ?

Il existe deux types de variables qui contiennent des dates dans SAS : toutes deux sont de type numérique. On distingue les simples dates et les « datetime ».

Les simples dates sont stockées comme des nombres de jours depuis le 1^{er} janvier 1960.

Les dates antérieures à 1960 sont stockées sous forme d'entiers négatifs. On peut stocker dans SAS des dates comprises entre le 1^{er} janvier 1582 (- 138 061) et le 31 décembre 20 000 (6 589 335).

Les datetime intègrent, comme leur nom l'indique, une dimension temporelle plus fine. Ils sont stockés en nombre de secondes depuis le 1^{er} janvier 1960 à minuit.

Comme pour les simples dates, tout ce qui est antérieur à 1960 est stocké comme un nombre négatif. Les datetimes sont compris entre le 1^{er} janvier 1582 minuit (- 11 928 470 400) et le 31 décembre 20 000 à 23h59 et 59 secondes (569 318 630 399).

On trouve également des variables numériques « time » qui ne contiennent que les informations en secondes.

Dates sous SAS

Les formats fournis par SAS

- Sans nécessiter d'intervention de l'utilisateur, une multitude de formats sont déjà créés dans SAS. Le tableau ci-dessous en récapitule une partie, parmi les plus utiles. Dans tous ces formats, le nombre qui précède immédiatement le point indique le nombre de caractères utilisés pour l'affichage. Dans le cas de données numériques, ce nombre de caractères inclut le séparateur décimal, un éventuel signe moins, un éventuel séparateur de milliers, etc.
- Si un nombre est situé après le point, il indique le nombre de décimales à afficher.

Données « caractère »		Dates SAS (nombres de jours depuis le 01/01/1960)	
\$2.	Ab	DDMMYY10.	24/11/2006
\$UPCASE2.	AB	MMYY57.	11/2006
Données numériques		YEAR4.	2006
5.	12345	FRADFWDX.	24 novembre 2006
7.2	1234.56	FRADFWDKX.	vendredi 24 novembre 2006
NUMX7.2	1234,56	FRADFMN.	novembre
NLNUM10.2	1 234,56	FRADFWDN.	Vendredi

- Les dates et datetime étant stockés en nombres de jours ou de secondes, leur affichage par défaut n'est pas très informatif.
Il convient donc de leur associer un format afin de les rendre plus lisibles.
- On peut ainsi, sans modifier ni créer de variable, agréger à l'année, au trimestre, au mois ou au jour de la semaine.

Quels formats utiliser sur des dates SAS ?

Principaux formats pour les simples dates

Nom du format	Exemple de valeur	Remarques
DDMMYY10.	25/01/2001	-
DDMMYYB10.	25 01 2001	-
DDMMYYD10.	25-01-2001	-
MMYYs7.	01/2001	-
YEAR4.	2001	-
FRADFwKX.	Jeudi 25 janvier 2001	-
FRADFwDX.	25 janvier 2001	-
FRADFmN.	janvier	-
FRADFwDN.	Jeudi	-
DATE9.	25JAN2001	Le mois est sur trois lettres en anglais (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC)
DAY2.	25	-
MONTH2.	1	-
QTRR3.	I	Trimestre en chiffres romains
WEEKDAY1.	5	Numéro du jour de la semaine (1=dimanche, 2=lundi, ...)
WEEKV.	2001-W04-04	Année 2001, semaine 4, 4 ^e jour (les conventions sont celles des numéros de semaines françaises, et le 1 ^{er} jour de la semaine est ici le lundi)
YYQRD9.	2001QI	Année et trimestre en chiffres romains

Nom du format	Exemple de valeur	Remarques
DATETIME.	02JAN60:17:40:00	
DATEAMPM.	02JAN60:05:40:00 PM	
DTDATE9.	02JAN1960	
DTMONYY7.	JAN1960	
DTYEAR4.	1960	
NLDATMW.	samedi 02 janvier 1960 17 h 40	Avec l'option système LOCALE=FRENCH
TOD10.1	17:40:00.0	TOD=Time Of Day

Quelles fonctions utiliser sur des dates SAS ?

Principales fonctions pour les simples dates

Fonction	Exemple d'utilisation	Résultat de l'exemple	Remarques
YEAR	YEAR (varDate)	2001	-
QTR	QTR (varDate)	1	1 ^{er} trimestre
MONTH	MONTH (varDate)	1	-
DAY	DAY (varDate)	25	-
WEEKDAY	WEEKDAY (varDate)	5	5 ^e jour de la semaine (début de la semaine = dimanche)
WEEK	WEEK (varDate, "V")	4	4 ^e semaine de l'année
TODAY	TODAY ()	17203	Date du jour (date système)
MDY	MDY (1, 25, 2001)	15000	-
INTCK	INTCK ("YEAR", 15000, TODAY ())	6	Nombre de 1 ^{er} jour de la période ¹ existant entre 2 dates
INTNX			-

Comment évoquer une date SAS dans un programme?

On peut utiliser différentes écritures, pour comparer une variable date avec le 25 janvier 2001 par exemple:

```
WHERE varDate > 15000 ;
```

```
WHERE varDate > "25jan2001"d ;
```

```
WHERE varDate > MDY(1,25,2001) ;
```

Le premier exemple est difficilement utilisable, à moins de connaître le nombre de jours depuis 1960 correspondant à la date à tester !

Pour le second, la norme est la suivante :

- Le nom du mois est sur 3 lettres, en anglais (**JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC**)
- La chaîne jour mois année est entre guillemets
- Un **D** est collé au guillemet fermant pour demander à SAS la conversion en nombre de jours depuis 1960.
- Majuscules et minuscules (tant sur le D que sur le nom du mois) sont équivalents.

Combinaison des tables

Superposition des tables: Instruction SET

Syntaxe

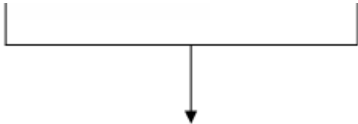
```
DATA _____;  
  SET ds1 ds2 ... ;  
  (BY var1 ... );  
  _____;  
RUN;
```

Exemple1

```
DATA Client_UPDATED;  
  SET Clients NEW_Clients;  
RUN;
```

	N_CLIENT	SEXE	COMPTE	PROFESSION	AGE	BIRTHDTE
1	1	M	12	Professeur	40	28DEC1970
2	2	F	13	Ingénieur	35	28DEC1975
3	3	F	15	Médecin	27	23FEV1983
4	4	M	10	Avocat	36	11FEV1974
5	5	F	9	Pilote	42	28DEC1968

	N_CLIENT	SEXE	COMPTE	PROFESSION	AGE	Etat_Familial
1	6	M	12	Profession libre	45	Marié
2	7	F	13	Instituteur	45	Marié



	N_CLIENT	SEXE	COMPTE	PROFESSION	AGE	BIRTHDTE	Etat_Familial
1	1	M	12	Professeur	40	28DEC1970	
2	2	F	13	Ingénieur	35	28DEC1975	
3	3	F	15	Médecin	27	23FEV1983	
4	4	M	10	Avocat	36	11FEV1974	
5	5	F	9	Pilote	42	28DEC1968	
6	6	M	12	Profession	45		Marié
7	7	F	13	Instituteu	45		Marié

Combinaison des tables

Superposition des tables: Instruction SET

Exemple 2

```
PROC SORT DATA=CLIENTS1;  
  BY AGE;  
RUN;
```

```
PROC SORT DATA=CLIENTS2;  
  BY AGE;  
RUN;
```

```
PROC SORT DATA=CLIENTS3;  
  BY AGE;  
RUN;
```

```
DATA COMBINED;  
  SET CLIENTS1 CLIENTS2 CLIENTS3;  
  BY AGE;  
RUN;
```

	N_CLIENT	SEXE	AGE
1	3	F	27
2	2	F	35
3	4	M	36
4	1	M	40
5	5	F	42

	N_CLIENT	SEXE	AGE
1	9	F	23
2	8	M	37

	N_CLIENT	SEXE	AGE
1	6	M	45
2	7	F	45



	N_CLIENT	SEXE	AGE
1	9	F	23
2	3	F	27
3	2	F	35
4	4	M	36
5	8	M	37
6	1	M	40
7	5	F	42
8	6	M	45
9	7	F	45

Les Procédures

- ❑ Une procédure de traitement des données est annoncée par le mot clé **PROC** suivi du nom de la procédure.
- ❑ Un certain nombre d'options et d'instructions permettent de définir le nom de la table en entrée, les variables à analyser, si les résultats sont stockés dans une autre table...
- ❑ Si aucun nom de table n'est indiqué après PROC et le nom de la procédure, SAS traite la dernière table créée.



Les Procédures

Proc Sort

Définition

Cette procédure est utilisée pour trier les observations d'une table SAS selon une ou plusieurs variables. Les observations, une fois triées, peuvent être stockées dans une nouvelle table (instruction OUT).

Syntaxe

```
PROC SORT DATA = _____ OUT= _____ NODUPREC | NODUPKEY ;  
    BY <DESCENDING> variable1 ..variablen ;  
RUN;
```

DATA = nom de la table SAS à trier.

OUT= nom de la table SAS de sortie. Indispensable si l'on ne veut pas que la table en sortie écrase la table en entrée.

noduprec élimine les observations redondantes.

nodupkey élimine les observations ayant la même valeur pour la clé de tri. C'est à-dire que seule la première occurrence d'une valeur de la variable de tri est conservée.

DESCENDING: Tri décroissant

Les Procédures

Proc Sort

Exemple

```
PROC SORT DATA=achat;  
  BY N_client;  
RUN;
```

AVANT TRI		
N_Client	Nb_vente	Montant
1	3	1
2	1	1
1	2	3
3	1	10
2	2	11
4	2	15
3	2	16
1	1	20
4	1	20
2	3	20

APRES TRI		
N_Client	Nb_vente	Montant
1	3	1
1	2	3
1	1	20
2	1	1
2	2	11
2	3	20
3	1	10
3	2	16
4	2	15
4	1	20

Les Procédures

Proc Print

Définition

La procédure Print permet d'imprimer tout ou partie d'une table SAS.

Syntaxe

```
PROC PRINT DATA = table [options];
```

<code>BY ;</code>	< sortie d'être effectuée pour chaque groupe.>
<code>ID ;</code>	<identifie l'observation.>
<code>VAR ;</code>	< liste les variables à imprimer.>
<code>SUM ;</code>	< liste les variables pour lesquelles des sommes sont calculées.>
<code>TITLE 'titre';</code>	
<code>LABEL ;</code>	<permet d'utiliser des labels pour les variables>.
<code>FORMAT ;</code>	<permet d'utiliser des formats sur les variables>.

```
RUN;
```

Les Procédures

Proc Print

Exemple

```
PROC PRINT DATA = vente noobs;  
  VAR ID_Client nb_vente mnt;  
  FORMAT mnt dollar4.;  
RUN;
```

ID_Client	nb_vente	mnt
001_001	1	\$11
001_001	2	\$10
001_002	1	\$6
001_002	2	\$7
001_002	3	\$4

Les procédures

Proc Format

Definition

La procédure Format permet de construire ses propres formats de lecture et d'écriture.

Un format permet de regrouper plusieurs modalités sous un même libellé. On distingue des formats caractères, appliqués à des variables caractères, et des formats numériques, appliqués à des variables numériques. Le format est un moyen de présenter (à l'affichage principalement) les données différemment de la façon dont elles sont physiquement stockées.

L'exemple le plus frappant est celui des dates : stockées par SAS comme un nombre de jours depuis le 1er janvier 1960, un format les affichera en jours, mois et années.

Dans une procédure statistique, l'utilisation d'un format peut éviter le recodage d'une variable.

Deux étapes :

- > *Création* et gestion de formats : **PROC FORMAT**
- > *Utilisation* de formats : instructions de **FORMAT** existants dans de nombreuses procédures.

Les procédures

Proc Format

Syntaxe

```
PROC FORMAT;  
VALUE nom_du_format  
Liste_de_valeurs='valeur1 formatée'  
.....  
Liste_de_valeurs='valeurn formatée' ;  
RUN;
```

Chaque liste de valeurs peut être composée

- d'une valeur simple : 75='Paris'
- d'une liste continue de valeurs : 92-94='Petite couronne'
- d'une liste exhaustive de valeurs : 92.93.94='Petite couronne'
- d'une liste discontinue de valeurs : 75.77.78.91-95='Région parisienne'

Les procédures

Proc Format

Exemple 1

Regroupement d'âges en classes :

```
PROC FORMAT;
VALUE $FQUIN 15-19='15 à 19 ans'
20-24='20 à 24 ans'
25-29='25 à 29 ans'
30-34='30 à 34 ans'
35-39='35 à 39 ans'
40-HIGH='40 ans et plus';
RUN;
```

Utilisation du format

```
PROC FREQ DATA=fic_age;
TABLES age;
FORMAT age $FQUIN.;
RUN;
```

Exemple 2

Regroupement des départements en région :

```
PROC FORMAT;
VALUE $REGION
'Paris','Hauts-de-Seine','Seine-Saint-
Denis','Val de
Marne','Yvelines',
'Val d'Oise','Seine et Marne'='Ile-de-
France'
'Ardennes','Aube','Marne','Haute-
Marne'='Champagne-Ardenne'
...
;
RUN;
```

Utilisation du format

```
PROC FREQ DATA=fic_dpt;
TABLES dpt;
FORMAT dpt $REGION.;
RUN;
```

Les procédures

Proc Transpose

Définition

Dans son expression la plus simple, la PROC TRANSPOSE transpose une table, c'est-à-dire transforme les lignes en colonnes.

La force de la Proc Transpose est de pouvoir ne transposer qu'une partie des observations.

La Proc Transpose ne génère aucune sortie dans l'Output

Syntaxe

```
PROC TRANSPOSE DATA=nom_table [options];
```

```
VAR liste_de_variables;
```

```
BY variable;
```

```
COPY liste_de_variables;
```

```
ID variable;
```

Les options :

L'option **out=** permet de définir une table en sortie différente de la table prise en entrée.

L'option **name=** permet de renommer la variable `_name_` dans le tableau transposé (variable qui contient le nom de la ou des variables transposées).

L'option **prefix=** permet de définir un préfixe pour les noms des nouvelles variables du tableau.

Les instructions :

L'instruction **VAR** fournit la liste des variables qui vont être transposées. Ces variables n'existent plus dans le tableau en sortie, et leurs valeurs sont réparties dans les cases appropriées du nouveau tableau.

L'instruction **BY** permet de définir l'identifiant de l'individu. Chaque modalité de cet identifiant correspondra à une ligne du tableau final.

L'instruction **ID** permet de définir la variable dont les modalités doivent définir les nouvelles variables du tableau.

L'instruction **COPY** définit les variables qui sont recopiées telles quelles dans le tableau final.

Les procédures

Proc Transpose

Exemple

Supposons que l'on dispose de la table notes suivante :

eleve	Matiere	note
tata	Math	15
tata	Français	10
titi	Math	14
titi	Français	18
toto	Math	8
toto	Français	12

eleve	_NAME_	math	fran_ais
tata	note	15	10
titi	note	14	18
toto	note	8	12

eleve	note_math	note_francais
tata	15	10
titi	14	18
toto	8	12

```
PROC SORT DATA=notes ; BY eleve ;
PROC TRANSPOSE DATA=notes out=notes2;
  VAR note;
  ID matiere;
  BY eleve;
RUN;
```

- L'instruction **BY** eleve permet bien d'obtenir une ligne par élève.
- L'instruction **ID** matiere définit bien la variable à transposer. La variable matiere n'existe plus, mais a été remplacée par autant de variables qu'il y avait de modalités de matiere. A noter que les caractères spéciaux (ici le ç) ne peuvent être traduits dans le nom de la nouvelle variable (ici il a été remplacé par un _).
- Enfin l'instruction **VAR** note définit le contenu des nouvelles cases : dans mon tableau croisé eleve*matiere, je souhaite mettre la note qui correspond à ce profil.
- Une colonne subsidiaire est créée, qui rappelle le nom de la variable que l'on a transposé.

Les Procédures

Proc Means

Définition

Cette procédure est la première qu'il faut connaître lorsqu'on souhaite effectuer des **statistiques descriptives** élémentaires sur des variables quantitatives. Par défaut, elle calcule le nombre d'observations non manquantes, la moyenne, l'écart-type, la valeur minimum et la valeur maximum de toutes les variables numériques de la table (ou des variables indiquées par l'instruction VAR). Avec les options appropriées, on peut demander un grand nombre de statistiques : somme, médiane, variance, skewness, kurtosis, quartiles, premier et dernier centile, premier et dernier décile, etc. Lorsqu'une (ou plusieurs) de ces options est spécifiée, cela annule l'édition par défaut. Il faudra donc expliciter toutes les statistiques que l'on souhaite obtenir.

Syntaxe

```
PROC MEANS DATA= table_entree [Mesures statistiques] [options] ;
  VAR liste_variables_quantitatives ;
  BY variable ;
  CLASS variable ; <CLASS définit les strates. Les variables énumérées dans l'instruction sont considérées
                    comme catégorielles >
  ID variable ;
  OUTPUT OUT=table_sortie;
Run;
```


Les Procédures

Proc Means

Les options :

Les options les plus intéressantes sont celles qui permettent de choisir les statistiques à éditer :

- N Effectif
- NMISS Nombre de valeurs manquantes
- MIN Minimum
- MAX Maximum
- RANGE Plage des valeurs = MAX – MIN
- SUMWGT Somme des poids
- SUM Somme
- MEAN Moyenne
- STD Ecart-type
- STDERR Standard Error of Mean
- KURTOSIS Coefficient d'aplatissement
- SKEWNESS Coefficient d'asymétrie
- USS Somme des carrés
- CSS Somme des carrés des écarts à la moyenne
- VAR Variance
- CV Coefficient de variation
- T Valeur de la statistique de Student pour le test (H0) : la moyenne est nulle
- PROBT P-value associée au test précédent
- MEDIAN Médiane
- QRANGE Distance interquartile = Q3 - Q1
- Q1 et Q3 Premier et troisième quartiles
- P1 P5 P10 P90 P95 P99 Centiles

Les Procédures

Proc Means

```
PROC MEANS DATA = Clients;  
RUN;
```

The MEANS Procedure				
Analysis Variable : AGE				
N	Mean	Std Dev	Minimum	Maximum
ff				
40	49.9500000	7.12588	21.0000000	77.0000000
ff				

Les Procédures

Proc Means

```
PROC SORT DATA=base; BY AGE SEXE;  
RUN;  
PROC MEANS DATA=base;  
BY AGE SEXE;  
RUN;
```

L'utilisation de BY dans Proc Means requiert des données triées

----- AGE= PLUS DE 50 ANS SEXE=F -----

The MEANS Procedure

Analysis Variable : MONTANT

N	Mean	Std Dev	Minimum	Maximum
10	34.9000000	5.9338951	27.0000000	43.0000000

----- AGE = 30-50 ANS SEXE=M -----

Analysis Variable : MONTANT

N	Mean	Std Dev	Minimum	Maximum
10	34.8000000	7.1460945	26.0000000	44.0000000

----- PROFESSION = MEDECIN SEX=F -----

CONTINUED....

Les Procédures

Proc Freq

Le propos de la procédure PROC FREQ est de faire des statistiques univariées ou bivariées sur des variables nominales. Elle permet donc de dresser des tableaux de fréquence et/ou des tableaux de contingence.

```
PROC FREQ DATA=nom_table;  
  TABLES listes_variables [options] ;  
  BY variable ;  
RUN;
```

Options: NOFREQ - NOPERCENT - NOROW -NOCOL -MISSING

L'instruction TABLES et ses options :

- L'instruction TABLES peut prendre deux formes :
- TABLES var1 var2 ... varn ; édite les tableaux de fréquence des variables var¹ jusqu'à varⁿ.
- TABLES var1*...*varn ; édite pour chaque profil de (var¹,...,varⁿ⁻²) le tableau croisé (tableau de contingence) de varⁿ⁻¹ par varⁿ.
- L'option out= permet d'enregistrer le résultat de l'instruction TABLES dans une table.

Les Procédures

Proc Freq

```
PROC FREQ DATA=clients;  
TABLES sexe;  
RUN;
```

The FREQ Procedure				
sexe	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	3	50.00	3	50.00
M	3	50.00	6	100.00

```
PROC FREQ DATA=clients;  
TABLES sexe*Profession;  
RUN;  
TITLE;
```

The FREQ Procedure				
Table of sexe by profession				
sexe	profession			
Frequency Percent Row Pct Col Pct	AVOCAT	INGENIEU R	MEDECIN	Total
F	2 33.33 66.67 100.00	1 16.67 33.33 50.00	0 0.00 0.00 0.00	3 50.00
M	0 0.00 0.00 0.00	1 16.67 33.33 50.00	2 33.33 66.67 100.00	3 50.00
Total	2 33.33	2 33.33	2 33.33	6 100.00

Les Procédures

Proc Freq

Exemple

```
proc freq DATA=malib.bidon;  
tables zone taille*zone ;  
run;
```

The FREQ Procedure						
zone	Frequency	Percent	Cumulative Frequency	Cumulative Percent		
IF	4	8.00	4	8.00		
NE	9	18.00	13	26.00		
NW	2	4.00	15	30.00		
SE	15	30.00	30	60.00		
SW	20	40.00	50	100.00		

Table of taille by zone						
taille		zone				
Frequency	Percent					
Row Pct	Col Pct	IF	NE	NW	SE	SW
1		0	4	0	11	18
		0.00	8.00	0.00	22.00	36.00
		0.00	12.12	0.00	33.33	54.55
		0.00	44.44	0.00	73.33	90.00
2		1	4	2	4	2
		2.00	8.00	4.00	8.00	4.00
		7.69	30.77	15.38	30.77	15.38
		25.00	44.44	100.00	26.67	10.00
3		3	1	0	0	0
		6.00	2.00	0.00	0.00	0.00
		75.00	25.00	0.00	0.00	0.00
		75.00	11.11	0.00	0.00	0.00
Total		4	9	2	15	20
		8.00	18.00	4.00	30.00	40.00

Exemple de commentaire :

40% des entreprises du secteur Bidon sont installées dans le sud ouest, ce qui représente 20 firmes.
90% de ces firmes sont des petites entreprises et 10% sont des PME.
Les grandes entreprises sont rares dans le secteur, puisqu'elles n'en représentent que 8%.

Les Fonctions

Les fonctions pour les variables numériques

Syntaxe: MEAN(argument, argument)

M1 = MEAN (4, 2, 3); → 3

Syntaxe: MAX(argument, argument)

Mx2 = MAX (1, ., 0); → 1

Syntaxe: INT(argument)

I1 = INT (-512.3); → -512

I2 = INT (33); → 33

Syntaxe SQRT(argument)

S1 = SQRT (15); → 3.87...

Syntaxe ROUND(argument, nearest)

R1 = ROUND (20.56, 1); → 21

R1 = ROUND (20.56, .5); → 20.5

R1 = ROUND (20.56, .1); → 20.6

Syntaxe ABS(argument)

A1 = ABS (-50.3); → 50.3

À suivre ...

- ❑ SAS Graph
- ❑ SQL
- ❑ SAS Macros



- ❑ & Mise à jour du support de cours

Références



Le site de SAS www.sas.com

<http://www.math.sciences.univ-nantes.fr/~lavancie/enseignement/PresentationSAS.pdf>

<http://www.uiowa.edu/~uisug/offerings.html>

www2.sas.com/proceedings/sugi22/TRAINING/PAPER319.PDF

www.agroparistech.fr/IMG/pdf/polySas.pdf

<http://helene-hamisultane.voila.net/travaux/SAS.pdf>

www.developpez.net/forums

Maitriser SAS BASE et SAS MACRO (SAS 9 et versions antérieures) : Hélène Kontchou Kouomegni et Olivier decourt

....