



# APPRENTISSAGE DU LANGAGE DE PROGRAMMATION SAS

Gramondo Martina

UT1 FOAD M2 & DU  
2018/19





SEMAINE 5:  
FORMATS & INFORMATS  
LES FONCTIONS  
IF THEN ELSE & SELECT WHEN

Gramondo Martina

UT1 FOAD M2 & DU  
2018/19



# FORMATS SAS

## INTRODUCTION

Un **format** est un attribut facultatif de la variable

Le format affiche le contenu de la variable dans une façon différente que sa vraie valeur stockée.

Le format ne modifie pas la valeur stockée de la table

Les formats sont utilisés:

- En phase de création et manipulation des données pendant la compilation pour intégrer dans une façon permanente cet attribut facultatif dans les métadonnées, dans ce cas on utilisera l'instruction d'affectation dans un étape DATA
- En phase de création d'un rapport des données pour afficher différemment la variable dans le résultat d'analyse dans une façon temporaire, dans ce cas on utilisera l'instruction d'affectation dans un étape PROC
- En phase de conversion avec la fonction INPUT, conversion entre variable caractère à numérique

Nous avons deux grandes familles des formats:

- Format SAS (date, heure, caractère, ...) sont des formats existants et il faudra les associer à la variable
- Format utilisateur, d'abord programmé par la PROC FORMAT et par la suite associé à la variable

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
...  
FORMAT variable nom_format. ;  
RUN;  
PROC ...;  
...  
FORMAT variable nom_format. ;  
RUN;
```

# FORMATS SAS

## INTRODUCTION

Un nom de **format** caractère suivre cette structure syntaxique:

**\$ nom\_format <w>.**

Un nom de **format** numérique suive cette structure syntaxique:

**nom\_format <w>.<d>**

FONCTION	DEFINITION
\$	Indique que le format s'applique à une variable caractère
nom_format	Nom propre du format SAS ou utilisateur
w	Specifie la longueur totale, le nombre des cases dédiées, à l'affichage du format (y compris les décimales, les caractères spéciaux, les séparateurs des millier, ...) Certains formats SAS et tous les formats utilisateurs ne necessitent pas de la longueur globale
.	Delimiteur obligatoire pour identifier que c'est un format
d	Pour les formats numeriques ou les devises, il specifie le nombre des decimales à afficher

# FORMATS SAS

## QUELQUE EXEMPLE FORMAT

Ici quelque exemple de format: caractère, numérique, date, heure

FORMAT	DEFINITION	FORMAT APPLIQUE	VALEUR STOCKE	VALEUR AFFICHE
\$w.	Format caractère pour écrire un nombre définit des caractères	\$4.	Programmation	Prog
w.d	Format numérique pour écrire des données numériques standard	7.1	12345.123	12345.1
EUROW.d	Format devise pour écrire des valeurs numériques avec un symbole euro (€) devant, une virgule comme séparateur de milliers et un point comme séparateur décimal	EURO10.2	12345.123	E12,345.12
EUROXw.d	Format devise pour écrire des valeurs numériques avec un symbole euro (€) devant, un point comme séparateur de milliers et une virgule comme séparateur décimal	EUROX10.2	12345.123	E12.345,12
DATEw.	Format date pour écrire la date avec: le jour sur 2 colonnes, le mois en 3 lettre en anglais et l'année sur 2 (DATE7.) ou 4 colonnes (DATE9.)	DATE9.	0	01JAN1960
DDMMYYw.	Format date pour écrire la date avec: le jour sur 2 colonnes, le mois sur 2 colonnes et l'année sur 2 (DDMMYY8.) ou sur 4 (DDMMYY10.) colonnes avec le séparateur /	DDMMYY10.	-1	31/12/1959
FRADFWDXw.	Format date pour écrire la date avec: le jour sur 2 colonnes, le mois en français et l'année sur 4 colonnes	FRADFWDX18.	1	2 janvier 1960

# FORMATS SAS

## QUELQUE EXEMPLE FORMAT

Si la largeur du format n'est pas assez grande pour afficher la valeur numérique sous la forme attendue, la valeur affichée est automatiquement ajustée pour convenir à la largeur

FORMAT APPLIQUE	VALEUR STOCKE	VALEUR AFFICHE	FORMAT APPLIQUE	VALEUR STOCKE	VALEUR AFFICHE
EUROX15.2	123456789.12	E123.456.789,12	EUROX8.2	123456789.12	1.23456E8
EUROX14.2	123456789.12	E123456789,12	EUROX7.2	123456789.12	1.235E8
EUROX13.2	123456789.12	E123456789,12	EUROX6.2	123456789.12	1.23E8
EUROX12.2	123456789.12	123456789,12	EUROX5.2	123456789.12	123E6
EUROX11.2	123456789.12	123456789,1	EUROX4.2	123456789.12	12E7
EUROX10.2	123456789.12	123456789	EUROX3.2	123456789.12	1E8
EUROX9.2	123456789.12	123456789	EUROX2.2	123456789.12	ERROR 156-185: The decimal specification
			EUROX1.2		ERROR 156-185: The decimal specification

# FORMATS SAS

## EXEMPLE

```
data class;  
montant=1500.1234;  
remise=300;  
remise_perc=remise/montant;  
date_aujourd'hui=today();  
code_postale=6100;  
format montant remise eurox9.2 remise_perc percent5.  
           date_aujourd'hui fradfwdx18. code_postale z5.;  
run;
```

Exemple

Associations permanente des  
formats aux variables via  
l'instruction **FORMAT**

**FORMAT** *variable nom\_format.* ;

Code standard

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)

# FORMATS SAS

## EXEMPLE

Format numérique **PERCENT5.**

Format numérique **Z5.**

Il utilise **5** cases pour afficher le numero, si le numero est moins long des 5 cases, ici 4, il complete avec des 0 à gauche

montant	remise	remise_perc	date_aujourd'hui	code_postale
E1.500,12	E300,00	20%	19 décembre 2018	06100

Format devise **EUROX9.2**  
il utilise 9

Format date **FRA DFWDX18.**  
**FRA** identifie la langue française

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)



# FORMATS UTILISATEUR

## INTRODUCTION

Un **format** peut être aussi créé par l'utilisateur.

Il faudra donc :

- Créer le format avec la PROC FORMAT
- Associer le format à la variable avec l'instruction FORMAT

Le nom d'un format utilisateur suit les règles syntaxiques suivantes :

- Il peut compter de 1 à 32 caractères
- Il doit commencer par une lettre ou un \_ (souligné du 8) ou **dans le cas d'un format caractère doit commencer par \$**
- Les caractères suivants (à partir du deuxième) peuvent être des lettres, des chiffres ou des \_
- **Il ne peut pas terminer par un chiffre**
- Vous pouvez mélanger la casse des lettres (majuscules et minuscules)
- SAS n'est pas sensible à la casse

```
PROC FORMAT;  
VALUE nom_format_u gamme='libellé'  
                        gamme='libellé'  
                        ...;  
  
RUN;  
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
FORMAT variable nom_format_u. ;  
RUN;  
PROC ...;  
  
...  
FORMAT variable nom_format_u. ;  
RUN;
```

# FORMATS UTILISATEUR

## INTRODUCTION

```
PROC FORMAT;  
VALUE nom_format_u gamme='libellé'  
                               gamme='libellé'  
                               ...;  
RUN;
```

Le **libellé d'un format** est la valeur que sera affiché quand la/les variables associées ont la valeur de la gamme

Le libellé:

- est toujours entre " (guillemets)
- Peut avoir jusqu'à 32767 caractères de long

La **gamme d'un format** est la liste des valeurs que faudra rencontrer dans la/les variables associées au format pour afficher le libellé

Le type de la gamme définira le type du format:

- Si un format à une gamme numérique alors le format sera un format numérique sinon sera un format caractère.

La gamme peut être:

- Une valeur unique
- Un intervalle des valeurs
- Une liste des valeurs

# FORMATS UTILISATEUR

## INTRODUCTION

```
PROC FORMAT;  
VALUE nom_format_u gamme='libellé'  
                               gamme='libellé'  
                               ...;  
RUN;
```

Dans la **gamme d'un format** est possible utiliser des mots clefs ainsi que des symboles

Les mots clefs possibles sont:

- OTHER: inclut toutes les valeurs qui ne correspondent à aucune autre valeur ou aucun autre intervalle
- HIGHT: substitue la plus grande valeur possible
- LOW :
  - pour les **formats numériques** substitue la plus petite valeur, **les missings sont exclus**
  - Pour les **formats caractères** substitue la plus petite valeur, **les missings sont inclus**

Les symboles pour les intervalles sont:

- X-Y: la valeur X et la valeur Y sont incluses de l'intervalle
- X-<Y: la valeur X est incluse dans l'intervalle, Y est exclues de l'intervalle
- X<-Y: la valeur X est exclue de l'intervalle et Y est incluse de l'intervalle
- X<-< Y: la valeur X et la valeur Y sont exclues de l'intervalle

# FORMATS UTILISATEUR

## EXEMPLE

Définition format **F\_AGE** format numérique et du format **\$GENRE** format caractère

Nom format ici **F\_AGE**

valeur de la gamme ici.

```
proc format;
value f_age
    .='age missing'
    low-12='age Q1'
    12<-13='age Q2'
    13<-15='age Q3'
    other='age Q4';
value $genre 'M'='Garçon'
    'F'='Fille'
    other='ERREUR';
run;
proc print data=sashelp.class;
var name age sex;
format age f_age. sex $genre.;
run;
```

Exemple

Libellé ici 'age missing'

Associations temporaires des formats utilisateurs aux variables via l'instruction **FORMAT**

```
PROC FORMAT;
VALUE nom_format_u gamme='libellé'
    gamme='libellé'
    ...;

RUN;
```

Code standard

# FORMATS UTILISATEUR

## EXEMPLE

Rapport créé avec la **PROC PRINT**

Obs.	Name	Age	Sex
1	Alfred	age Q3	Garçon
2	Alice	age Q2	Fille
3	Barbara	age Q2	Fille
4	Carol	age Q3	Fille
5	Henry	age Q3	Garçon
6	James	age Q1	Garçon
7	Jane	age Q1	Fille
8	Janet	age Q3	Fille
9	Jeffrey	age Q2	Garçon
10	John	age Q1	Garçon
11	Joyce	age Q1	Fille
12	Judy	age Q3	Fille
13	Louise	age Q1	Fille
14	Mary	age Q3	Fille
15	Philip	age Q4	Garçon
16	Robert	age Q1	Garçon
17	Ronald	age Q3	Garçon
18	Thomas	age Q1	Garçon
19	William	age Q3	Garçon

Format numérique utilisateur **F AGE.**  
Associé à la variable **AGE**

Format caractère utilisateur **\$GENRE.**  
Associé à la variable **SEX**

# INFORMATS

## INTRODUCTION

Un **informat** est un attribut facultatif de la variable

Le **informat** permet de interpréter correctement la valeur d'une variable, et donc de la stocker correctement.

Les **informats** sont utilisés:

- En phase de importation des fichier pour interpréter la valeur brute correctement et la stocker dans une façon correcte pour SAS. Les **informat**s permettront aussi de déduire le type et la longueur en phase de compilation pendant un programme de importation des fichiers
- En phase de exportation d'une table SAS dans une autre système de fichier des donnée, pour construire les métadonnées de la table de sortie
- En phase de conversion avec la fonction PUT, conversion entre variable numérique à caractère

Nombreux sont les nom des **FORMATS** que on retrouve dans les **INFORMATS**

# INFORMATS

## INTRODUCTION

Un **informat** applicable à une variable caractère, on l'écrira:

**\$ nom\_informat <w>.**

Un **informat** applicable à une variable numérique, on l'écrira:

**nom\_informat <w>.**

FONCTION	DEFINITION
\$	Indique que le informat s'applique à une variable caractère
nom_informat	Nom propre du informat SAS ou utilisateur
w	Spécifie la largeur totale de lecture (y compris les décimales, les caractères spéciaux, les séparateurs des millier, ...) Selon le nom du format et pour les formats utilisateurs n'est pas requis
.	Délimiteur obligatoire du nom du format SAS

NB: Dans le cas d'un INFORMAT on ne spécifiera pas le d, le nombre des décimales

# INFORMATS

## QUELQUE EXEMPLE FORMAT

Ici quelque exemple de informat: caractère, numérique, date, heure

INFORMAT	DEFINITION	INFORMAT APPLIQUE	VALEUR BRUTE	VALEURE STOCKEE
\$w.	Informat caractère pour lire un nombre définit des caractères	\$13.	programmation	programmation
w.	Informat numérique pour lire des données numériques standard	9.	12345.123	12345.12
EUROW.d	Informat devise pour lire des valeurs numériques avec un symbole euro (€) devant, une virgule comme séparateur de milliers et un point comme séparateur décimal	EURO10.	E12,345.12	12345.12
EUROXw.d	Informat devise pour lire des valeurs numériques avec un symbole euro (€) devant, un point comme séparateur de milliers et une virgule comme séparateur décimal	EUROX10.	E12.345,12	123451.12
DATEw.	Informat date pour lire la date avec: le jour sur 2 colonnes, le mois en 3 lettre en anglais et l'année sur 2 (DATE7.) ou 4 colonnes (DATE9) et l'interprètera comme une date SAS	DATE9.	01JAN1960	0
DDMMYYw.	Informat date pour lire la date avec: le jour sur 2 colonnes, le mois sur 2 colonnes et l'année sur 2 (DDMMYY8.) ou sur 4 (DDMMYY10.) colonnes avec le séparateur / et l'interprètera comme une date SAS	DDMMYY10.	31/12/1959	-1
PERCENTw.	Informat numérique pour lire un valeur avec des entier, des décimales séparés par un . Et le symbole % et interpréter la valeur numérique sur SAS	PERCENT5.	56.12%	0.5612



# INFORMATS LES DÉCIMALES

## Dans l'écriture d'un informat

- si les décimales ne sont pas spécifiées SAS arrivera à les interpréter même au delà du nombre des décimales spécifiées dans d
- par contre si on impose une longueur de décimales et SAS ne le trouve pas dans la valeur alors il divisera la valeur par d pour les calculer.

INFORMAT APPLIQUE	VALEUR BRUTE	VALEUR STOCKEE
EUROX8.	E123.456	123456
EUROX8.2	E123.456	1234.56
EUROX8.2	E123,45	123.45
EUROX8.2	E123,456	123.456

valeur entière sans décimales il lit correctement la valeur

**On impose 2 décimales** et quand **il ne les trouve pas** il va les calculer ici il fait **123456/100**

il lit correctement les valeurs et quand il y a plus que 2 décimales il va les lire comme même

# FONCTIONS SAS

## INSTRUCTION

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
Nom_variable=nom_fonction(param1, param2, ..);  
WHERE nom_fonction(param1, param2, ..)=valeur;  
IF nom_fonction(param1, param2, ..)=valeur;  
...  
RUN;
```

Une **fonction** est une **routine** qui **retourne une valeur déterminée en fonction d'argument(s)**

La **fonction travaille par ligne**, càd que pour chaque boucle de l'exécution (**\_N\_**) la fonction nous permettra d'avoir un résultat

Nous pouvons utiliser des fonctions:

- Dans une instruction d'affectation de la valeur d'une variable
- Dans un WHERE ou dans un IF dans la condition pour filtrer les données

Les fonctions SAS sont classées en fonction du type de manipulation des données à exécuter :

- *Caractères*
- *Dates et heures*
- *Statistiques*
- *Mathématiques*
- ...

# FONCTIONS SAS

## INSTRUCTION

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
Nom_variable=nom_fonction(param1, param2, ..);  
WHERE nom_fonction(param1, param2, ..)=valeur;  
IF nom_fonction(param1, param2, ..)=valeur;  
...  
RUN;
```

La syntaxe classique d'une **fonction SAS** est la suivante:

- Chaque fonction a un **nom propre** suivie des parenthèses ()
- En présence des **paramètres**, il faudra les écrire dans **l'ordre demandé** par la syntaxe de la fonction
- En présence des **plusieurs paramètres** il faudra les **séparer** par des ,
- En présence des **paramètres mot clé** il faudra les écrire entre “
- Souvent les fonctions ont des paramètres facultatifs (dans l'help la notation est la suivante <parametre\_facultatif>)

Quand la **fonction** est **utilisée comme instruction d'affectation à une variable**, le type du résultat de la fonction permettra à SAS en phase de **compilation** de **deduire le type de la variable et la longueur** (souvent les fonctions caracteres affectent 200 octées à la variable resultat)

Une **imbrication des fonctions** est l'utilisation des fonctions dans les paramètres d'une autres fonction, la fonction la plus imbriquée sera résolue pour première.

L'utilisation de l'help de SAS est indispensable quand on utilise des fonctions

# FONCTIONS SAS

## EXEMPLE

La fonction est utilisée pour initialiser la variable **NBJ\_RENTREE**  
**INTCK** est une fonction date donc le resultat sera une variable  
numerique de 8 octets **N8**

Fonction **TODAY ()** pas de parametres demandés  
Ici **TODAY ()** est une fonction imbriquée car la valeur  
servira comme parametre à la fonction **INTCK ()**

```
data class;  
Date_rentree="03SEP2018"d;  
Nbj_rentree= intck('day', date_rentree, today());  
run;
```

Exemple

Nom\_fonction(paramètre, 'parametre mot-cléf')

Code standard

Fonction **INTCK** a 3 parametres obligatoires dont un mot clef  
Syntaxe de la fonction:

**INTCK('unité de mesure', date\_ancienne, date\_recente)**

unité de mesure (**day, month, semiyear, ...**) il faudra le mettre entre ''

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)

# FONCTIONS SAS

## QUELQUE FONCTION DATE

Les fonctions Date permettent de manipuler des variables numerique particulieres, càd des variables date SAS (un nombre de jours entre la date de référence (01/01/1960) et la date en question)

Ici quelque exemple des fonctions pour extraire des informations à partir d'une date SAS:

FONCTION	DEFINITION
TODAY()	Demande la date SAS au système
YEAR(date)	Extrait l'année d'une date SAS et renvoie l'année sur quatre chiffres
QTR(date)	Extrait le trimestre d'une date SAS et renvoie un nombre de 1 à 4
MONTH(date)	Extrait le mois d'une date SAS et renvoie un nombre de 1 à 12
DAY(date)	Extrait le jour du mois d'une date SAS et renvoie un nombre de 1 à 31
WEEKDAY(date)	Extrait le jour de la semaine d'une date SAS et renvoie un nombre de 1 à 7, où 1 représente le dimanche

# FONCTIONS SAS

## QUELQUE FONCTION DATE

Ici quelque exemple des fonctions pour manipuler des dates SAS:

FONCTION	DEFINITION
INTCK('unité mesure', date_start, date_stop, '<alignement>')	Returne la difference entre deux dates SAS dans l'unité de mesure demandée, on peut aligner les dates
INTNX('unité mesure', date_start, increment, '<alignement>')	Incremente une date SAS avec un intervalle dans l'unité de mesure demandée, on peut allinier les dates
YRDIF(date_start, date_end, basis)	Returne la difference entre deux dates SAS en années, le type d'année est basis
DATDIF(date_start, date_stop, basis)	Returne la difference entre deux dates SAS en jours, le type d'année est basis
MDY(mois,jour,année)	Renvoie une valeur de date SAS à partir des valeurs numériques que seront affectés au mois, au jour et à l'année

NB: dans la syntaxe des fonctions <> indique que le parametre est facultatif

# FONCTIONS SAS

## QUELQUE FONCTION DATE: EXEMPLE

```
data class;  
*date_fin_ecole=date fin construction de l'ecole;  
date_fin_ecole="10JUN1990"d;  
date_aujourd'hui=today();  
age_batiment_1=intck("year", date_fin_ecole, today());  
age_batiment_2=yrdif(date_fin_ecole, today(), 'ACT/ACT');  
anniversaire_2018=mdy(month(date_fin_ecole), day(date_fin_ecole), year(today()));  
anniversaire_50_ans=intnx('year', date_fin_ecole, 50, 'sameday');  
*Format date_fin_ecole date_aujourd'hui anniversaire_50 date9.;  
run;
```

NB: L'instruction FORMAT est en commentaire afin de montrer que on travail bien sur des dates SAS, toutefois l'application d'un format rendra plus compréhensible les valeurs des dates

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)

# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE: EXEMPLE

Calcule de la différence entre deux dates SAS en années 'YEAR' entre la DATE\_FIN\_ECOLE et la date SAS du jour  
**intck("year", date\_fin\_ecole, today());**

Constante date "10JUN1990"d  
pour récupérer la date en date SAS

Calcule de la date SAS que sera basée sur **DATE\_FIN\_ECOLE** plus **50** années 'YEAR'  
Plusieurs sont les options pour gérer le jours et le mois de la nouvelle date SAS, ici on demande le même jour 'SAMEDAY'  
**intnx('year', date\_fin\_ecole, 50, 'sameday');**

Obs.	date_fin_ecole	date_aujourd'hui	age_batiment_1	age_batiment_2	anniversaire_2018	anniversaire_50_ans
1	11118	21538	28	28.5288	21463	29381

Fonction **TODAY ()** pour récupérer la data  
Du jour en date SAS **13/12/2018**

Calcule de la différence entre deux dates SAS en années avec plusieurs choix possible de paramétrer l'année (ici "ACT/ACT" année réelle)  
**yrdif(date\_fin\_ecole, today(), 'ACT/ACT');**

Construction d'une date SAS avec 3 variables numériques, le mois et le jour de DATE\_FIN\_ECOLE ainsi que l'année de la date SAS d'aujourd'hui  
**mdy(month(date\_fin\_ecole), day(date\_fin\_ecole), year(today()));**



# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE

Les fonctions caracteres permettent de manipuler des variables caracteres.

Ici quelque exemple des fonctions pour harmoniser la casse des variable caractère:

FONCTION	DEFINITION
UPCASE(var_caractere)	Convertir toutes les lettres de la variable en majuscule
LOWCASE(var_caractere)	Convertir toutes les lettres de la variable en minuscule
PROPCASE(var_caractere, '<delimiteur>')	Convertir la premiere lettre de chaque mot de la variable en majuscule les autres minuscule

NB: dans la syntaxe des fonctions <> indique que le parametre est facultatif

# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE

Ici quelque exemple des fonctions pour extraire des souchaines de caractere dans une variable caractere:

FONCTION	DEFINITION
SUBSTR(var_caractere, colonne_debut, <nb_colonnes>)	Extraction à partir de la colonne debut, jusqu'à la fin ou pour un nombre des colonnes indiqué
SCAN(var_caractere, ennieme_mot, '<delimiteur>')	Extraction de ennieme mot, delimiteur des mots par default sinon à definir

NB: dans la syntaxe des fonctions <> indique que le parametre est facultatif

# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE

Ici quelque exemple des fonctions pour chercher et chercher/remplacer des souchaines de caractere dans une variable caractere:

FONCTION	DEFINITION
FIND(var_caractere, 'souchaine_recherchée', <modificateur>, <colonne_debut_recherche>)	Extraction à partir de la colonne debut, jusqu'à la fin ou pour un nombre des colonnes indiqué
TRANWRD(var_caractere, 'souchaine_recherchée', 'souchaine_remplacement')	Remplace ou enleve toutes les occurrence d'un mot donné dans une variable

NB: dans la syntaxe des fonctions <> indique que le parametre est facultatif

# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE

Ici quelque exemple des fonctions pour concatener des variables caracteres :

FONCTION	DEFINITION
CAT(var_caractere,..., var_caracteren)	Concatene les variables caracteres en parametre, aucune gestion des blancs
CATT(var_caractere,..., var_caracteren)	Concatene les variables caracteres en parametre, il enleve les espaces à droite
CATS(var_caractere,..., var_caracteren)	Concatene les variables caracteres en parametre, il enleve les espaces à droite et gauche
CATX('delimiteur', var_caractere, ..., var_caracteren)	Concatene les variables caracteres en parametre, il enleve les espaces à droite et à gauche et il ajoute après chaque variable le delimiteur passé en parametre

NB: dans la syntaxe des fonctions <> indique que le parametre est facultatif

# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE: EXEMPLE

```
data class;  
*correction nom_prenom;  
nom_prenom_original='Alfred, steFANO';  
nom_prenom=upcase(tranwrd(nom_prenom_original,'FA','PHA'));  
*solution pas à pas;  
nom=scan(nom_prenom,-1,' ');  
prenom_1=scan(nom_prenom,1,' ');  
prenom_2=substr(prenom_1,1,1);  
p_nom=catx('.', prenom_2, nom);  
p_nom_2=propcase(p_nom, '.');  
*solution avec imbrication des fonctions;  
p_nom_3= propcase(catx('.',substr(scan(nom_prenom,-1,' '),1,1), scan(nom_prenom,-1,' ')), '.');  
run;
```

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)

# FONCTIONS SAS

## QUELQUE FONCTION CARACTERE: EXEMPLE

Recupere la première mot, celui avant le ','

```
scan(nom_prenom, 1, ',')
```

Première lettre en majuscule et le restant en minuscule, le delimitateur des mots est ','

```
propcase(p_nom, ' ');
```

Concatenation de l'initial prenom avec '.' avec ne nom de famille

```
catx('.', prenom_2, nom);
```

Recupere la première lettre du prenom

```
substr(prenom_1, 1, 1)
```

Chercher FA et le remplacer avec PHA

```
tranwrd(nom_prenom_original, 'FA', 'PHA')
```

Fonction pour mettre le resultat tout en majuscule

```
UPCASE(tranwrd(nom_prenom_original, 'FA', 'PHA'))
```

Obs.	nom_prenom_original	nom_prenom	nom	prenom_1	prenom_2	p_nom	p_nom_2	p_nom_3
1	AlfreD, steFANO	ALFRED, STEPHANO	STEPHANO	ALFRED	A	A.STEPHANO	A.Stephano	S.Stephano

# CONVERSION

## CONVERSION AUTOMATIQUE

Dans une variable SAS on peut:

- Changer le nom de la variable (RENAME instruction ou option de table)
- Changer la longueur de la variable (LENGTH en instruction)
- Mais on ne peut pas changer un type de variable

Faire une conversion est utiliser dans un type de variable pas conforme à ce que est attendu

Ils existent deux types de conversions:

- Automatique: SAS voir l'incompatibilité du type de variable et essaye de faire la conversion.
- Explicite: SAS n'arrive pas à faire la conversion automatique et il faut l'aider avec les fonctions

La conversion automatique est faisable quand:

- Dans la conversion d'une variable caractère à une numérique quand la valeur est un valeur standard, càd une valeur compréhensibles par SAS (valeur entier, valeur avec . comme séparateur décimales, valeur négatif, exponentiel,...)
- Dans la conversion d'une variable numérique en caractère il faudra toujours gérer les espaces à gauches dû à l'alignement

# CONVERSION

## CONVERSION AUTOMATIQUE: EXEMPLE

```
data class;  
id_etudiant_c='112';  
nouveau_id_etudiant_n=id_etudiant_c+10000;  
prix_c='150';  
commission='10';  
nouveau_prix_n=sum(prix_c*0.9,commission);  
id_etudiant_3_c=catx('-', 'EDU',nouveau_id_etudiant_n);  
run;
```

NOTE: Character values have been converted to numeric values at the places given by: (Line):(Column).

76:23 79:20 79:31

NOTE: The data set WORK.CLASS has 1 observations and 6 variables.

NOTE: DATA statement used (Total process time):

real time 0.00 seconds

cpu time 0.00 seconds

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)



# CONVERSION

## CONVERSION AUTOMATIQUE: EXEMPLE

Dans une fonction SUM SAS nécessite des variables numériques  
La conversion automatique est faite avec succès pour **150** et **10**  
`sum(prix_c*0.9,commission) ;`

variable caractère

Obs.	id_etudiant_c	nouveau_id_etudiant_n	prix_c	commission	nouveau_prix_n	id_etudiant_3_c
1	112	10112	150	10	145	EDU-10112

Dans une expression arithmétique SAS nécessite des variables numériques  
Ici prend la valeur **112** caractère et essaye de la convertir en numérique  
Le résultat est pertinent et donc écrit un WARNING et fait le calcul  
`id_etudiant_c+10000;`

Dans une fonction de la famille de la concaténation SAS nécessite des variables caractères  
Si il trouve des numériques il faut la conversion automatique et la gestion des espaces à gauche  
`catx('-', 'EDU',nouveau_id_etudiant_n) ;`

# FONCTIONS SAS

## FONCTIONS CONVERSION EXPLICITE

Dans le cas où la conversion automatique ne peut pas marcher car les données ne sont pas des données standard (ex: dates, devises, pourcentages, ...) ou pour éviter le message de conversion automatique dans le Journal, on peut utiliser la conversion explicite avec les fonctions:

FONCTION	DEFINITION
INPUT(var_caractere, informat)	Utilise la valeur de la variable caractere comme si était une variable numerique, il va l'interpreter via l'informat
PUT(var_numerique, format)	Utilise la variable numerique comme si était une variable caractere, il va l'afficher via le format

# FONCTIONS SAS

## FONCTIONS CONVERSIONS EXPLICITE: EXEMPLE

```
data class;
nom_prenom_c='Alfred, Stefano';
date_naissance_c='18/06/1989';
*solution pas à pas;
date_naissance_n=input(date_naissance_c, ddmmyy10.);
age_n=int(yrdif(date_naissance_n, '31DEC2018'd, 'ACT/ACT'));
nom_c=upcase(scan(nom_prenom_c,-1,','));
id_etudiant_1_c=catx('-', 'ETU', nom_c, put(age_n, z5.));
*solution avec imbrication des fonctions;
id_etudiant_2_c=catx('-', 'ETU', upcase(scan(nom_prenom_c, 2, ',')),
                    put(yrdif(input(date_naissance_c, ddmmyy10.),
                    '31DEC2018'd, 'ACT/ACT'), z5.));
run;
```

NB: Dans cette étape DATA il n'y a pas de instruction SET car elle est composée que des constantes initialisée dans l'étape data  
Le programme sera executé 1 fois seulement (\_N\_=1)

# FONCTIONS SAS

## FONCTIONS CONVERSIONS EXPLICITE: EXEMPLE

Difference entre date naissance numerique et une constante date  
**yrdif(date\_naissance, '31DEC2018'd, 'ACT/ACT')**  
Il gardent que la parties entiere avec **INT**

Concaténation du préfixe **ETU** avec nom famille et age  
sur 6 colonnes chaque champs délimité par -

**catx('-', 'ETU', nom\_c, put(age\_n, z5.))**  
La concatenation accept que des variables caracteres il  
faut aussi faire la conversion de Q de numerique au  
character

Obs.	nom_prenom_c	date_naissance_c	date_naissance_n	age_n	nom_c	id_etudiant_1_c	id_etudiant_2_c
1	Alfred, Stefano	18/06/1989	10761	29	STEFANO	ETU-STEFANO-00029	ETU-STEFANO-00030

Date naissance caractere  
(alignée à gauche)

Date SAS suite à une conversion de caractère à  
numerique avec un informat **DDMMYY10.**  
**input(date\_naissance\_c, ddmmyy10.)**

Récupération du nom de famille  
**scan(nom\_prenom\_c, -1, ' ', 'P')**  
Premiere letter majuscule et les autres  
minuscule **PROPCASE**

Concatenation de l'initial prenom avec '.'  
avec ne nom de famille

**catx('.', prenom\_2, nom);**

# TRAITEMENT CONDITIONNEL

## IF THEN ELSE

Dans l'instruction IF nous allons mettre des expression:

- Si l'expression est vérifiée l'instruction suivant le THEN sera exécutée
- Si l'expression n'est pas vérifiée l'instruction après le ELSE sera exécutée

Dans l'instruction THEN on écrira une seule action à exécuter

Si les instructions sont plus nombreuses il faudra les encapsuler dans les instructions DO et END

Dans l'instruction ELSE on pourra avoir des actions à faire ou un nouveau control ELSE IF...

L'expression est composé d'un opérande et d'un operateur

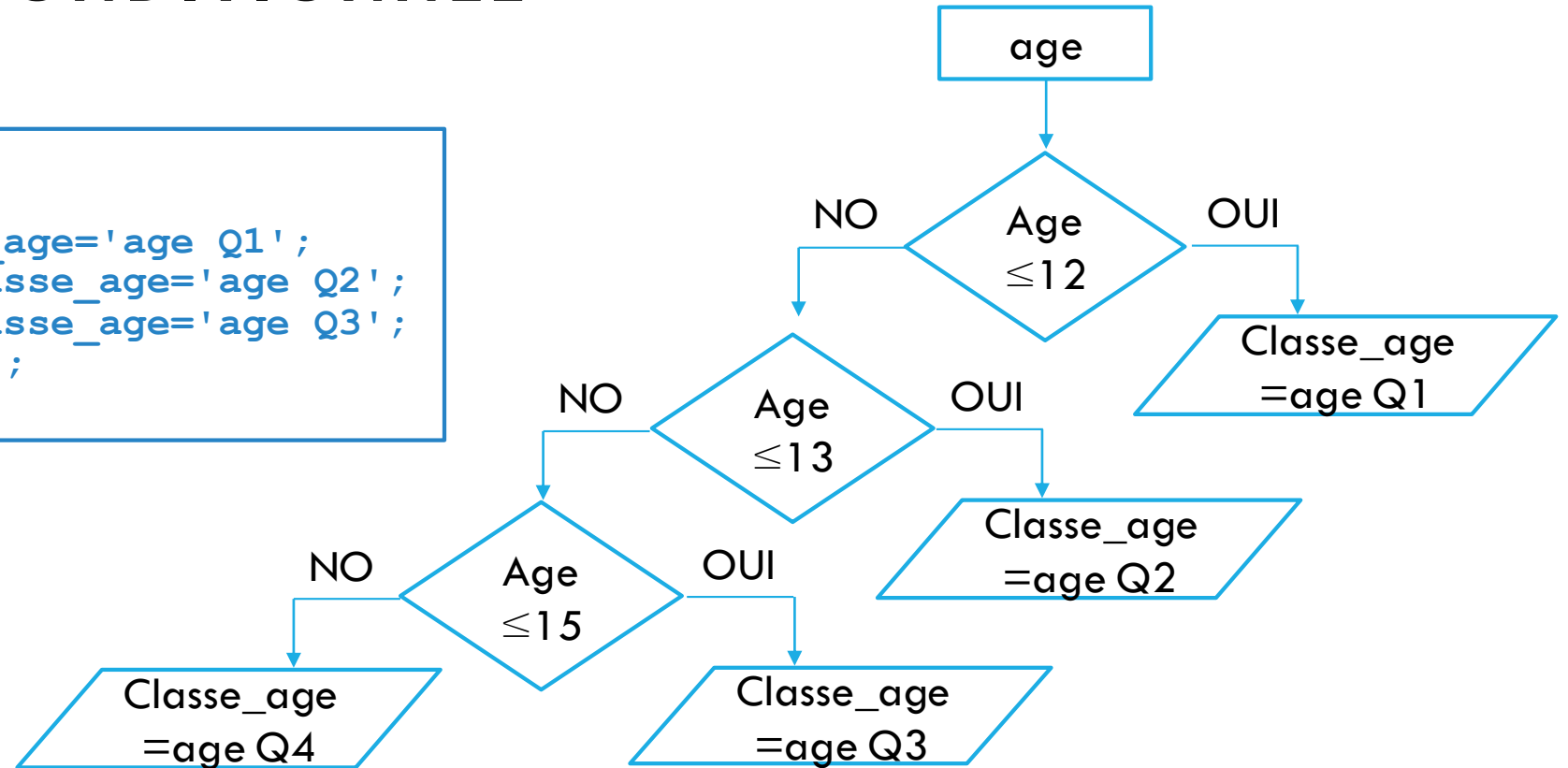
Les operateurs exclusifs du WHERE ne pourront donc pas être utilisés (like, contains, is null, is missing, between and).

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
IF operande operateur THEN  
    instruction;  
ELSE instruction;  
IF operande operateur THEN DO;  
    instruction;  
    instruction;  
END;  
ELSE DO; instruction;  
        instruction;  
END;  
RUN;
```

# TRAITEMENT CONDITIONNEL

## IF THEN ELSE

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```



# COMPILATION

## EXEMPLE FIN COMPILATION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE
\$ 8	\$ 1	N 8	N 8	N 8	\$8



Bloque descripteur des variables de WORK.CLASS

NOM	TYPE	LONG
NAME	\$	8
SEX	\$	1
AGE	N	8
HEIGHT	N	8
WEIGHT	N	8
CLASSE_ETUDIANT	\$	8

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
		.	.	.	.	1	0



# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

Instruction **SET** permet de  
charger dans le PDV  
la première ligne de  
**sashelp.class**

### Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

### Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alfred	M	14	69.0	112.5	.	1	0

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

FAUX

Condition **if age<=12** pas vérifiée  
on cherche l'instruction **else**

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alfred	M	14	69.0	112.5	.	1	0

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

FAUX

Condition **if age<=13** pas vérifiée  
on cherche l'instruction **else**

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alfred	M	14	69.0	112.5	.	1	0

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

VRAIS


Condition **if age<=15** est vérifiée  
L'instruction après le **then** sera exécutée  
Ici **classe\_age='age Q3'**;

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alfred	M	14	69.0	112.5	Age Q3	1	0

# EXÉCUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION



```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

### Zone de données de WORK.CLASS

Alfred	M	14	69.0	112.5	Age Q3
--------	---	----	------	-------	--------

### Program Data Vector



NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alfred	M	14	69.0	112.5	Age Q3	1	0

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alfred	M	14	69.0	112.5		2	0

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

Instruction **SET** permet de  
charger dans le PDV  
la deuxième ligne de  
**sashelp.class**

### Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

### Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alice	F	13	56.5	84.0	.	2	0

# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

FAUX

Condition **if age<=12** pas vérifiée  
on cherche l'instruction **else**

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alice	F	13	56.5	84.0	.	2	0



# EXECUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

VRAIS


Condition **if age<=13** est vérifiée  
L'instruction après le **then** sera exécutée  
Ici **classe\_age='age Q2'**;

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alice	F	13	56.5	84.0	Age Q2	2	0

# EXÉCUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION



```
data class;  
set sashelp.class;  
if age <=12 then classe_age='age Q1';  
else if age<=13 then classe_age='age Q2';  
else if age<=15 then classe_age='age Q3';  
else classe_age='age Q4';  
run;
```

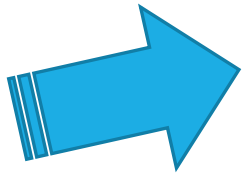
Sortie implicite  
Retour implicite



### Zone de données de WORK.CLASS

Alice	F	13	56.5	84.0	Age Q2
-------	---	----	------	------	--------







### Program Data Vector



NAME	SEX	AGE	HEIGHT	WEIGHT	CLASSE_AGE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	\$8	N 8	N 1
Alice	F	13	56.5	84.0	Age Q2	2	0

```
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time           0.01 seconds
```


**Fenêtre  
Journal**

Colonnes	
<input checked="" type="checkbox"/>	Sélectionner tout
<input checked="" type="checkbox"/>	 Name
<input checked="" type="checkbox"/>	 Sex
<input checked="" type="checkbox"/>	 Age
<input checked="" type="checkbox"/>	 Height
<input checked="" type="checkbox"/>	 Weight
<input checked="" type="checkbox"/>	 classe_age

## Fenêtre Données en Sortie

# EXÉCUTION

## EXEMPLE INSTRUCTION PAR INSTRUCTION



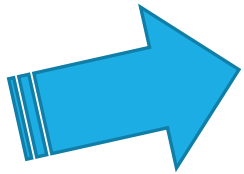
```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Fin exécution `_N_=1`  
Sortie implicite, le PDV écrit  
l'observation 1 dans la table  
**WORK.CLASS**  
Retour implicite, on recommence  
une autre boucle, donc `_N_=2`

### Zone de données de WORK.CLASS

Alfred	M	14	69.0	112.5	College	19	21430	36000	91	Alfred
--------	---	----	------	-------	---------	----	-------	-------	----	--------

### Program Data Vector



NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	<b>_N_ N 8</b>	<b>_ERROR_ N 1</b>
Alfred	M	14	69.0	112.5	College	19	21430	36000	91	Alfred	1	0

# TRAITEMENT CONDITIONNEL

## IF THEN ELSE IMBRIQUE

```
data class;  
set sashelp.class;  
*plusieurs instructions après THEN;  
priorite=0;
```

```
if sex='F' then do;
```

```
  if age<=12 then do;
```

```
    classe_etudiant='F age Q1';
```

```
    priorite=1;
```

```
  end;
```

```
  else if age <=13 then classe_etudiant='F age Q2';
```

```
  else if age <=14 then classe_etudiant='F age Q3';
```

```
  else classe_etudiant='F age Q4';
```

```
end;
```

```
if sex='M' then do;
```

```
  if age<=12 then do;
```

```
    classe_etudiant='M age Q1';
```

```
    priorite=1;
```

```
  end;
```

```
  else if age <=13 then classe_etudiant='M age Q2';
```

```
  else if age <=15 then classe_etudiant='M age Q3';
```

```
  else classe_etudiant='M age Q4';
```

```
end;
```

```
run;
```

Le première principale sur le **sex**

Le deuxième contrôle est sur l'âge en rapport à première contrôle sur **sex**  
C'est comme si on avait:

**If sex='F' and age<=12**

Le contrôle principale sur le **sex**

Le deuxième contrôle est sur l'âge en rapport à première contrôle sur **sex**  
C'est comme si on avait:

**If sex='M' and age<=12**

# ETAPE DATA RESULTAT

NOTE: There were 19 observations read from the data set SASHELP.CLASS.  
NOTE: The data set WORK.CLASS has 19 observations and 8 variables.  
NOTE: DATA statement used (Total process time):  
    real time            0.00 seconds  
    cpu time             0.00 seconds

Fenêtre  
Journal

CODE

JOURNAL

RESULTATS

DONNEES EN SORTIE

Table :

WORK.CLASS

Afficher :

Libellés de la colonne

Filtrer : (néant)

Colon

<

Fenêtre  
Données en  
Sortie

# EXERCICES

## SEMAINE 5



KEEP  
CALM  
AND  
DO YOUR  
HOMEWORK

- Déclarer la bibliothèque UT1\_SAS qui pointe sur votre fichier (pour moi UT1\_FOAD)  
`libname UT1_SAS '/folders/myfolders/UT1_FOAD' ;`
- A l'afficher le bloque descripteur de la table UT1\_SAS.EMPLOYEE\_PAYROLL
- A partir de la table UT1\_SAS.EMPLOYEE\_PAYROLL avec la PROC FREQ contrôler les modalités de variables EMPLOYEE\_GENDER MARITAL\_STATUS DEPENDENTS
- A l'aide de la PROC FORMAT créer les formats utilisateurs suivants:
  - \$GENRE avec les correspondances suivantes: M=Garçon, F=Fille, erreur de saisie dans les autres cas
  - \$STATUS\_M avec les correspondances suivantes: S: Célibataire/ Divorcé, M=Marié, O=Autres situations, ' '=Missing
- Créer une table appelée EMPLOYEE\_PAYROL dans la WORK à partir de la table UT1\_SAS.EMPLOYEE\_PAYROLL (ajouter l'instruction LENGTH)

```
data employee_payroll;  
set ut1_foad.employee_payroll;  
length rappeler_employee_promotion $ 50;  
run;
```

- Création variable DUREE\_COLLABORATION que sera la différence en années entre la date de embauche (EMPLOYEE\_HIRE\_DATE) et la date de départ (EMPLOYEE\_TERM\_DATE)
- Création variable RAPPELER\_EMPLOYEE que prendra la valeur:
  - « Salarié de l'entreprise »: quand la date de terme du contrat est missing (EMPLOYEE\_TERM\_DATE)
  - « Ex-salarié de l'entreprise: Oui rappeler en priorité »: quand la DUREE\_COLLABORATION est inferieur à 5 ans
  - « Ex-salarié de l'entreprise: Oui rappeler »: quand la DUREE\_COLLABORATION est entre 6 et 20 ans
  - « Ex-salarié de l'entreprise: Non ne pas rappeler »: quand la DUREE\_COLLABORATION est supérieure à 20 ans

# EXERCICES

## SEMAINE 5

- Création variable PROMOTION que prendra la valeur:
  - « Ajouter un collaborateur à son équipe »: quand la DUREE\_COLLABORATION est supérieur à 20 ans et le nombre des DEPENDENTS est inferieur à 5
  - « Pas de promotion »: pour les collaborateurs toujours dans l'entreprise
  - « Pas pertinent: Ex-collaborateur »: pour les ex-collaborateurs
- Associer les formats d'affichages dans une façon permanente (formats utilisateurs créés dans la PROC FORMAT), formats date9. pour les dates sauf pour BIRTH\_DATE où sera affiché seulement l'année YEAR4.
- Afficher la table via un PROC PRINT, ici le résultats des premières lignes:



KEEP  
CALM  
AND  
DO YOUR  
HOMEWORK

Obs.	Employee_ID	Employee_Gender	Salary	Birth_Date	Employee_Hire_Date	Employee_Term_Date	Marital_Status	Dependents	rappeler_employee	promotion	duree_collaboration
1	120101	Garçon	163040	1976	01JUL2003	.	Celibataire/divorcé	0	Salarié de l'entreprise	Pas de promotion	16
2	120102	Garçon	108255	1969	01JUN1989	.	Autres situations	2	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	30
3	120103	Garçon	87975	1949	01JAN1974	.	Marié	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
4	120104	Fille	46230	1954	01JAN1981	.	Marié	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	38
5	120105	Fille	27110	1974	01MAY1999	.	Celibataire/divorcé	0	Salarié de l'entreprise	Pas de promotion	20
6	120106	Garçon	26960	1944	01JAN1974	.	Marié	2	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
7	120107	Fille	30475	1949	01FEB1974	.	Marié	2	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
8	120108	Fille	27660	1984	01AUG2006	.	Celibataire/divorcé	0	Salarié de l'entreprise	Pas de promotion	13
9	120109	Fille	26495	1986	01OCT2006	.	Marié	3	Salarié de l'entreprise	Pas de promotion	13
10	120110	Garçon	28615	1949	01NOV1979	.	Marié	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	40
11	120111	Garçon	26895	1949	01NOV1974	.	Marié	3	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
12	120112	Fille	26550	1969	01JUL1990	.	Marié	3	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	29
13	120113	Fille	26870	1944	01JAN1974	.	Celibataire/divorcé	0	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
14	120114	Fille	31285	1944	01JAN1974	.	Marié	3	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
15	120115	Garçon	26500	1984	01AUG2005	.	Marié	2	Salarié de l'entreprise	Pas de promotion	14
16	120116	Garçon	29250	1959	01FEB1980	.	Celibataire/divorcé	0	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	39
17	120117	Garçon	31670	1964	01APR1986	.	Autres situations	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	33
18	120118	Garçon	28090	1959	01JUL1984	.	Marié	3	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	35
19	120119	Garçon	30255	1969	01JAN1998	.	Marié	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	21
20	120120	Fille	27645	1944	01JAN1974	.	Marié	3	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
21	120121	Fille	26600	1944	01JAN1974	.	Marié	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	45
22	120122	Fille	27475	1954	01JUL1978	.	Celibataire/divorcé	0	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	41
23	120123	Fille	26190	1964	01OCT1985	31JAN2005	Marié	3	Ex-salarié de l'entreprise: Oui rappeler	Pas pertinent: Ex-collaborateur	20
24	120124	Garçon	26480	1959	01MAR1979	.	Marié	1	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	40
25	120125	Garçon	32040	1954	01MAR1979	31JUL2004	Marié	2	Ex-salarié de l'entreprise: Non ne pas rappeler	Pas pertinent: Ex-collaborateur	25
26	120126	Garçon	26780	1988	01AUG2006	.	Autres situations	2	Salarié de l'entreprise	Pas de promotion	13
27	120127	Fille	28100	1979	01NOV1998	.	Marié	2	Salarié de l'entreprise	Ajouter un collaborateur à son équipe	21
28	120128	Fille	30890	1986	01NOV2006	.	Celibataire/divorcé	0	Salarié de l'entreprise	Pas de promotion	13
29	120129	Garçon	30070	1964	01OCT1985	31MAR2003	Celibataire/divorcé	0	Ex-salarié de l'entreprise: Oui rappeler	Pas pertinent: Ex-collaborateur	18
30	120130	Garçon	26955	1984	01MAY2006	.	Marié	2	Salarié de l'entreprise	Pas de promotion	13



# EXERCICES

## SEMAINE 5



**KEEP  
CALM  
AND  
DO YOUR  
HOMEWORK**

- A partir de la table WORK.EMPLOYEE\_PAYROLL créer la table PROMOTION avec seulement les collaborateurs ayant droit à une promotion (PROMOTION=« Ajouter un collaborateur à son équipe »)
- A l'aide de l'instruction KEEP et sélectionner seulement les variables suivantes: EMPLOYEE\_ID, DEPENDENTS, DUREE\_COLLABORATION, EMPLOYEE\_HIRE\_DATE

CODE JOURNAL RESULTATS DONNEES EN SORTIE			
Table :	WORK.PROMOTIONS	Afficher :	Noms de colonnes
Filtrer : (néant)			
Colonnes			
Lignes totales : 192 Colonnes totales : 4			
Lignes 1-100			
<input checked="" type="checkbox"/> Sélectionner tout	Employee_ID	Employee_Hire_Date	Dependents
<input checked="" type="checkbox"/> Employee_ID			duree_collaboration
<input checked="" type="checkbox"/> Employee_Hire_Date			
<input checked="" type="checkbox"/> Dependents			
<input checked="" type="checkbox"/> duree_collaboration			
Propriété	Valeur		
Libellé			
Nom			
Longueur			
Type			
Format			
Informat			
1	120102	01JUN1989	2
2	120103	01JAN1974	1
3	120104	01JAN1981	1
4	120106	01JAN1974	2
5	120107	01FEB1974	2
6	120110	01NOV1979	1
7	120111	01NOV1974	3
8	120112	01JUL1990	3
9	120113	01JAN1974	0
10	120114	01JAN1974	3
11	120116	01FEB1980	0
12	120117	01APR1986	1
13	120118	01JUL1984	3
14	120119	01JAN1998	1
15	120120	01JAN1974	3
16	120121	01JAN1974	1
17	120122	01JUL1978	0
18	120124	01MAR1979	1
19	120127	01NOV1998	2
20	120132	01OCT1978	0
21	120143	01OCT1982	1

# EXERCICES

## SEMAINE 5



**KEEP  
CALM  
AND  
DO YOUR  
HOMEWORK**

- BONUS: Discrétiser la variable SALARY en quantiles via une nouvelle variable
- Utiliser la [PROC UNIVARIATE](#) sur la table WORK.EMPLOYEE\_PAYROLL et analyser la variable SALARY pour connaître les valeurs des quartiles (nombreuses sont les sorties de cette procédure, il y aura aussi les informations sur les quartiles)
- Créer une table appelée EMPLOYEE\_PAYROL\_SALARY dans la WORK à partir de la table WORK.EMPLOYEE\_PAYROLL (elle n'est pas indispensable mais vous pouvez ajouter [l'instruction LENGTH](#))

```
data employee_payroll_salary;  
set employee_payroll;  
length salary_q $ 15;  
run;
```

- Création variable SALARY\_Q que prendra la valeur:
  - « Salary Q1 » jusqu'à « Salary Q4 » pour identifier les tranches de la variable SALARY
- A partir de la table EMPLOYEE\_PAYROL\_SALARY utiliser la PROC FREQ pour contrôler la distribution de la nouvelle variable SALARY\_Q
- Ici le résultat:

La procédure FREQ

salary_q	Fréquence	Pourcentage	Fréquence cumulée	Pourcentage cumulé
Salary Q1	106	25.00	106	25.00
Salary Q2	106	25.00	212	50.00
Salary Q3	106	25.00	318	75.00
Salary Q4	106	25.00	424	100.00

# EXERCICES

## SEMAINE 5



KEEP  
CALM  
AND  
DO YOUR  
HOMEWORK

- BONUS: Créer un format à partir des quartiles de la variable SALARY
- Avec l'aide d'une PROC FORMAT créer un format utilisateur Q\_SALARY avec les 4 classes des quartiles plus une classe missing (les valeurs sont issue de la PROC UNIVARIATE)
- Associer le format Q\_SALARY à la variable SALARY dans une façon permanente dans la table crée précédemment WORK. EMPLOYEE\_PAYROL\_SALARY
- Avec l'aide de la PROC MEANS et de la PROC FREQ sur la table WORK.EMPLOYEE\_PAYROL\_SALARY et la variable SALARY constater la versatilité d'un format en rapport au recodage via la création d'une nouvelle variable et faire une PROC MEANS et une PROC FREQ sur la variable SALARY
- Ici les résultats:

La procédure MEANS

Variable d'analyse : Salary				
N	Moyenne	Ec-type	Minimum	Maximum
424	38041.51	31741.14	22710.00	433800.00

La procédure FREQ

Salary	Fréquence	Pourcentage	Fréquence cumulée	Pourcentage cumulé
Salary Q1	106	25.00	106	25.00
Salary Q2	106	25.00	212	50.00
Salary Q3	106	25.00	318	75.00
Salary Q4	106	25.00	424	100.00

NB: Pour la PROC UNIVARIATE et de la PROC MEANS faudra juste utiliser l'option pour spécifier la table à analyser (DATA) et l'instruction pour spécifier la variable d'analyse (VAR), pas besoin d'ajouter des autres options ni instructions.



## REPONSES DANS LE FORUM

Proposées vos réponses dans le forum, vous disposez aussi de ma proposition de correction.

Lien pour l'help SAS:

[Doc SAS](#)