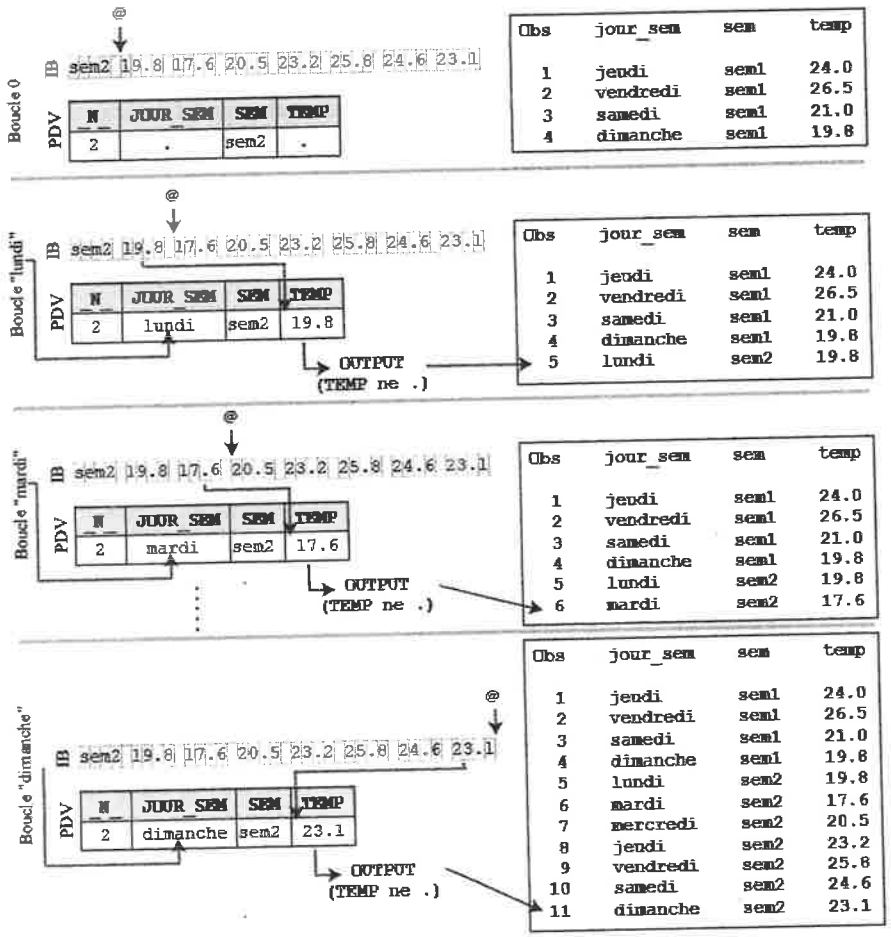

Figure 2.4 • Principe de fonctionnement du programme 2.21 – traitement du second enregistrement.

2.4.2 Plusieurs enregistrements pour une observation

Vous allez apprendre dans cette section à transformer plusieurs enregistrements de votre fichier en une observation. Si vous passez par un CARDS, dans quelques rares cas, il se peut que vous n'ayez pas besoin d'outils de programmation spécifique[1]. En revanche, si vous devez construire une table SAS à partir d'un fichier TXT extérieur à SAS, il vous faudra nécessairement utiliser des options d'INFILE spécifiques lorsque les données relatives à une même observation seront présentées sur plusieurs lignes du fichier de données.

a. Cas simple : une observation construite au moyen de x enregistrements

Imaginons que nous ayons dans un fichier les données suivantes à transformer en table SAS :

```
----+----1----+----2----+----3----+----4----+
Letters from Iwo Jima
Lettres d'Iwo Jima
2006  8.1
Ken Watanabe
Flags of Our Fathers
Mémoires de nos pères
2006  7.2
Ryan Phillippe
```

Sur la base de ce que vous savez déjà, sans les outils de programmation que nous développons plus loin, vous pourriez écrire le programme 2.22.

Programme 2.22

```
DATA clint3;
INPUT titre_us $ 1-21 titre_fra $ 1-21 annee note acteur $ 1-14;
CARDS;
... les données...
;
```

Ce programme ne fonctionne pas notamment parce que SAS ne peut pas comprendre que TITRE_US est à lire dans le premier enregistrement (entre les colonnes 1 et 21) et que TITRE_FRA est aussi à lire entre les colonnes 1 à 21, mais du second enregistrement.

La compréhension du premier champ ne doit pas poser de problème, mais il faut indiquer à SAS qu'il doit « charger » un nouvel enregistrement pour lire la modalité de la seconde variable. Vous pouvez à cet effet utiliser le programme 2.23.

Programme 2.23

```
DATA clint3;
   INPUT titre_us $ 1-21;
   INPUT titre_fra $ 1-21;
   INPUT annee note;
   INPUT acteur $ 1-14;
CARDS;
... les données...
;
```

Programme 2.24

```
DATA clint3;
INPUT titre_us $ 1-21 /
      titre_fra $ 1-21 /
      annee note /
      acteur $ 1-14;
CARDS;
... les données...
;
```

Vous avez aussi la possibilité d'utiliser l'argument '/' de l'instruction INPUT (programme 2.24). Le résultat que vous obtenez est parfaitement équivalent.

Les données utilisées ici sont reprises dans le fichier CLINT3.TXT. Le programme 2.25 crée une table à partir de ce fichier.

Programme 2.25

```
DATA clint3;
   INFILE "C:\intro_SAS\fichiers\clint3.txt" TRUNCOVER;
   INPUT titre_us $ 1-21 / titre_fra $ 1-21 / annee note / acteur $ 1-14;
RUN;
```

Vous êtes obligé de passer par un TRUNCOVER parce que les titres n'ont pas tous 21 caractères. Vous pouvez également recourir au pointeur #X. Votre ligne INPUT devient :

```
NPUT titre_us $ 1-21 #2 titre_fra $ 1-21 #3 annee note #4 acteur $ 1-14;
```

Si vous utilisez un pointeur #X, SAS regarde la valeur maximale qu'il prend dans l'instruction INPUT. Il en déduit que, pour construire une observation, il devra gérer plusieurs enregistrements. Dans le premier enregistrement, il observera la modalité de TITRE_US, et dans le quatrième, la modalité de ACTEUR.

Tout comme le pointeur @X permet de se déplacer à droite et à gauche dans un enregistrement, le pointeur #X permet de revenir en arrière.

```
IPUT titre_us $ 1-21 #2 titre_fra $ 1-21 #3 annee note #4 acteur $ 1-14 #1 mot $;
```

Le programme demande à SAS, une fois la variable ACTEUR saisie, de charger à nouveau dans l'INPUT BUFFER la première ligne, afin de construire une variable avec le premier mot du titre. Le fait d'utiliser le pointeur #X vous autorise donc des « retours en arrière » que ne permet pas '/'.

Exercice 2.8 : *Dans le fichier SGTPEPPER.TXT, vous trouverez la liste des chansons qui composent l'album Sgt Pepper's Lonely Hearts Club Band, des Beatles. Le fichier est organisé de la manière suivante :*

```
----+----1----+----2----+----3----+----4----+
1  Sgt. Pepper's Lonely Hearts Club Band
Lennon  McCartney  2:02  A
```

Ordre / titre de la chanson /

Compositeur 1 / compositeur 2 / durée du titre / face

Créez une table SAS à partir de ce fichier. Vous devez utiliser le fichier externe – ne faites pas de copier-coller de vos données dans l'éditeur.

b. Un premier cas de données hiérarchisées

Dans la section précédente, les modalités des variables d'une observation apparaissaient certes dans plusieurs enregistrements, mais on pouvait compter sur le fait que chaque observation provenait d'un même nombre d'enregistrements. Les données dont nous nous servirons pour le programme 2.26 diffèrent sensiblement par leur organisation.

En effet, on n'observe pas la même régularité que dans le cas précédent. Un premier enregistrement indique le nom de la famille à considérer (clé 1) et les suivants (autant d'enregistrements que de membres dans la famille), les clés 2, précisent le prénom et l'âge des membres de la famille. Ce type de données hiérarchisées est par exemple typique des recensements de population.

Programme 2.26

```
DATA famille;
   RETAIN nom;
   INPUT cle @;
   IF cle=1 THEN INPUT @11 nom $;
   ELSE DO;
      INPUT @3 prenom $12. @15 age;
      OUTPUT;
   END;
CARDS;
1 famille Dupont
2 Pierre      38
2 Sophie      37
2 Théodore     6
2 Marcelline   5
1 famille Dubois
2 Frédéric    42
2 Michelle    37
2 Marie       11
1 famille Durand
2 Jean        53
1 famille Dufour
2 Hervé       78
2 Marie       75
;RUN;
```

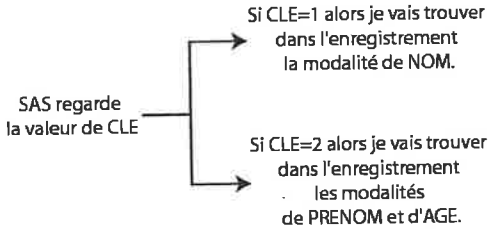

Figure 2.5 • Principe de fonctionnement du programme 2.26.

Le programme 2.26 fait appel à des éléments de programmation que nous n'avons pas encore vus. Si vous lisez cet ouvrage pour la première fois, vous pourrez revenir plus tard sur ce programme et ses commentaires.

Nous résumons à la figure 2.5 le principe du programme 2.26. Dans ce programme, nous indiquons *via* RETAIN (voir section 3.6.1) que SAS doit conserver d'une observation à l'autre, dans le PDV (voir section 3.1), la modalité de la variable NOM.

L'instruction INPUT qui suit permet d'enregistrer la modalité de CLE. Nous utilisons une nouvelle fois @ pour demander à SAS « d'attendre » et donc de ne pas vider l'INPUT BUFFER. Nous allons en fait introduire des traitements conditionnés à la valeur de CLE.

Si CLE=1, alors SAS saisit, *via* une nouvelle instruction INPUT, une variable caractère NOM, à lire à partir de la 11e colonne.

Sans l'option @, la seconde instruction INPUT aurait ordonné le chargement d'un nouvel enregistrement (seconde ligne de données) et la modalité de NOM aurait alors été égale à « 38 ».

L'instruction ELSE indique que, dans le cas contraire, les instructions situées entre les instructions DO (voir section 3.5.1) et END (DO s'arrête avec END) sont à exécuter. SAS saisira alors PRENOM et AGE, puis il écrira, dans la table, l'observation (instruction OUTPUT) composée des modalités prises par NOM (qu'il a en mémoire dans le PDV), PRENOM et AGE.

at 2.8

```
 nom      cle    prenom       age

Dupont     2    Pierre        38
Dupont     2    Sophie        37
Dupont     2    Théodore       6
Dupont     2    Marcelline     5
Dubois     2    Frédéric      42
Dubois     2    Michelle      37
Dubois     2    Marie         11
Durand     2    Jean          53
Dufour     2    Hervé         78
Dufour     2    Marie         75
```

Exercice 2.9 : Le fichier BEATLES.TXT a une structure très proche de celle de SGTPEPPER.TXT, auf qu'il s'agit ici de traiter non plus un, mais quatre albums des Beatles.

```
---1----+----2----+----3----+----4----+
 Sgt. Pepper's Lonely Hearts Club Band
. Pepper's Lonely Hearts Club Band
  McCartney  2:02  A
```

a ligne 0 donne l'année de sortie de l'album ainsi que son titre ; les lignes suivantes (>0) reprennent lans le même ordre, et avec la même structure, le numéro d'ordre de la chanson, son titre puis, sur a ligne suivante, le premier compositeur, le deuxième compositeur, la durée (minutes, secondes) et la ace sur le trente-trois tours original. Vous devez obtenir la table suivante, dont voici un extrait :

```
                                                duree duree
album        annee  num   titre     compo1  compo2     M     S    face
Abbey Road    1969   8    Because   Lennon  McCartney  2     45     B
```

. Un second cas de données hiérarchisées

maginons que les enregistrements soient maintenant organisés comme ceci :

```
----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+
lle  Dupont
'e       38  Sophie       37  Théodore      6  Marcelline    5
lle  Dubois
'ric     42  Michelle     37  Marie        11
.le  Durand
         53
.le  Dufour
'        78  Marie        75
```

ci, vous souhaitez combiner les deux premiers enregistrements pour obtenir dans otre table quatre observations (une par membre de la famille) : les enregistrements 3 t 4 produiront trois observations, etc. Dans ce cas, c'est la compréhension de la struc- ure du fichier qui vous aidera à trouver la solution. On remarque déjà que la clé est ci égale au nombre de membres de la famille et que les enregistrements, pour une ême valeur de clé, sont organisés en colonnes. Pour chaque clé > 0, le premier rénom commence en colonne 3, le deuxième en colonne 18, le troisième en olonne 33 et le quatrième en colonne 48. Le programme 2.27 prend en compte ces iverses informations pour construire correctement la table souhaitée.

Programme 2.27

```
DATA famille (DROP=cle i);
   RETAIN nom;
   INPUT cle @;
   IF cle=0 THEN INPUT @11 nom $;
   ELSE DO i=1 TO cle;
      INPUT @(3+15*(i-1)) prenom $12. age @;
      OUTPUT;
   END;
CARDS;
... les données ...
;RUN;
```

Résultat 2.9

Obs	nom	prenom	age
1	Dupont	Pierre	38
2	Dupont	Sophie	37
3	Dupont	Théodore	6
4	Dupont	Marcelline	5
5	Dubois	Frédéric	42
6	Dubois	Michelle	37
7	Dubois	Marie	11
8	Durand	Jean	53
9	Dufour	Hervé	78
10	Dufour	Marie	75

À nouveau, le programme 2.27 fait appel à des outils de programmation que nous n'avons pas encore vus. La principale différence avec le programme 2.26 réside dans la boucle DO (voir section 3.5.2), qui va « tourner » autant de fois que la famille a de membres (information donnée par CLE). Lorsque I=1 (premier tour de boucle), l'information PRENOM est à lire en colonne $3 = 3 + 15 \times (1 - 1)$. C'est ensuite la variable AGE qui est lue. L'option @ demande d'une part à SAS de ne pas passer à l'enregistrement suivant et d'autre part que d'autres traitements sont possibles par la suite. Les deux variables ont maintenant leur modalité ; l'instruction OUTPUT indique à SAS qu'il doit écrire dans la table les modalités des variables NOM, PRENOM et AGE. Examinons à présent cette même sous-partie du programme pour I=2. La variable PRENOM est à lire à partir de la 18e colonne ($3 + 15 * (2 - 1)$), AGE vient ensuite…

Cette configuration des données n'est pas commune. Ayez cependant à l'esprit que vous pourrez toujours construire une table SAS fidèle à vos données, quelle que soit leur organisation, si vous avez compris la logique de cette organisation.

Il est aussi important de saisir qu'un même lot de données peut conduire à des tables différentes. Pour ce nouvel exemple, nous modifions l'organisation des données utilisées dans le programme 2.27.

Programme 2.28

```
DATA famille2 (KEEP=nom membre);
   INFILE CARDS MISSOVER;
   RETAIN nom;
   INPUT cle @;
   IF cle=1 THEN INPUT @11 nom $;
   membre=0;
   IF cle=2 THEN DO UNTIL (age=.);
      INPUT prenom $ age @;
      IF age=. THEN OUTPUT;
      membre+1;
   END;
CARDS;
1 famille Dupont
2 Pierre 38 Sophie 37 Théodore 6 Marcelline 5
1 famille Dubois
2 Frédéric 42 Michelle 37 Marie 11
1 famille Durand
2 Jean 53
1 famille Dufour
2 Hervé 78 Marie 75
;RUN;
```

Résultat 2.10

Obs	nom	membre
1	Dupont	4
2	Dubois	3
3	Durand	1
4	Dufour	2

L'objet de ce programme est de construire une table dans laquelle vous aurez, à côté de chaque nom, le nombre de membres que comprend la famille (voir résultat 2.10). Par rapport au programme 2.27, vous notez la présence d'une boucle DO UNTIL. Celle-ci exécutera diverses instructions jusqu'à ce que AGE soit égal à valeur manquante : lorsque AGE est valeur manquante, cela indique que l'enregistrement de type 2 est entièrement lu. Les conditions spécifiées dans les boucles DO UNTIL sont évaluées en bas de la boucle, ce qui assure que les tâches demandées à l'intérieur de la boucle seront effectuées au moins une fois. En fait, elles le seront autant de fois que la famille compte de membres. Le comptage des membres est réalisé par une variable incrémentée MEMBRE, initialisée à zéro avant que la boucle ne débute. Nous illustrons à la figure 2.6 le fonctionnement de ce programme sur les données relatives à la famille Dubois.

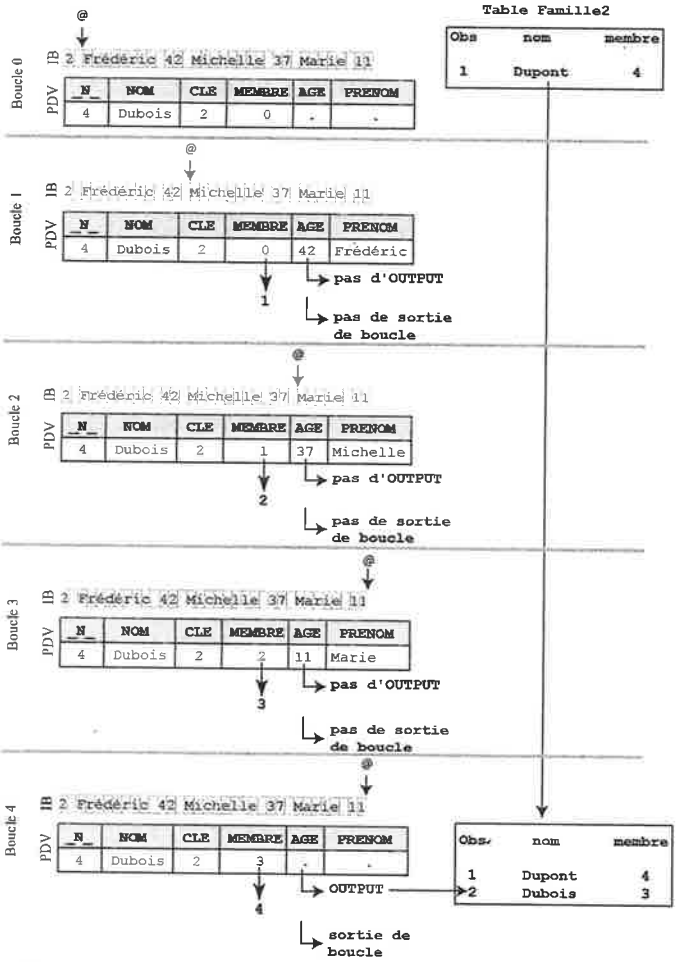

Figure 2.6 • Illustration du fonctionnement du programme 2.28.

Le traitement des deux premiers enregistrements a permis la création de la première observation de la table FAMILLE2. Le traitement du troisième enregistrement s'est traduit par ce que vous observez sur la figure à l'étape BOUCLE 0 :

- le positionnement du curseur de lecture des données dans l'INPUT BUFFER après le champ contenant la modalité de CLE ;
- l'attribution à la variable NOM de la modalité « Dubois » (variable de type RETAIN).

Avant que la boucle DO UNTIL débute, on donne à MEMBRE la modalité 0, AGE et PRENOM sont en valeur manquante (puisque ces deux variables ne sont pas de type RETAIN).

Lorsque débute la boucle (étape boucle 1 de la figure), l'instruction INPUT permet la lecture du prénom puis de l'âge du premier membre de la famille. Le curseur de lecture des données dans l'INPUT BUFFER stoppe sa lecture immédiatement après l'espace qui suit l'âge de Frédéric. Puisque l'âge n'est pas valeur manquante, alors il n'y a pas de versement du PDV dans la table FAMILLE2. MEMBRE est ensuite augmenté de 1 (et vaut donc 1), et puisque l'âge n'est pas égal à valeur manquante, on ne sort pas de la boucle.

Pour les boucles 2 et 3, le fonctionnement est identique : on enregistre PRENOM et AGE. Ensuite, comme AGE n'est pas valeur manquante, il n'y a pas versement du PDV dans la table à créer, et MEMBRE – au départ égal à la valeur donnée au tour de boucle précédent – est augmenté de 1.

Au départ de la boucle 4, le curseur de lecture des données est en fin de ligne (après l'âge de Marie). Comme nous avons interdit à SAS de regarder sur l'enregistrement suivant *via* l'option MISSOVER d'INFILE, les modalités de PRENOM et d'AGE sont mises en valeurs manquantes. Le contenu du PDV (en fait, uniquement les variables citées dans l'option de table KEEP= de l'instruction DATA) est versé dans la table. MEMBRE est augmenté de 1, mais comme l'instruction OUTPUT a déjà été exécutée, la modalité « 4 » n'apparaîtra pas par la suite. À nouveau, puisque AGE est valeur manquante, on sort de la boucle.

Nous arrivons donc « en bas du programme » pour ce quatrième enregistrement. Le programme va maintenant traiter le cinquième enregistrement…

Rappelons que vous pouvez passer cette section si vous lisez cet ouvrage pour la première fois, et qu'il convient, avant lecture, de bien maîtriser les concepts que nous exposerons au chapitre 3. Mais si vous avez déjà lu et travaillé le chapitre 3, ce nouvel exemple de fonctionnement du PDV doit vous convaincre (si ce n'est déjà fait) qu'il est absolument nécessaire de comprendre le fonctionnement interne de SAS afin de maîtriser la programmation à mettre en place pour le traitement de cas particuliers.

Exercice 2.10 : *À partir des données utilisées par le programme 2.27, reconstruisez une table identique à celle obtenue par le biais du programme 2.28 sans utiliser le fait que la clé est égale au nombre de membre que compte la famille. (exercice difficile).*

Exercice 2.11 : *Si vous avez travaillé le chapitre 3, réécrivez ce programme en utilisant une boucle DO WHILE plutôt qu'une boucle DO UNTIL. Pourquoi devez-vous absolument introduire une instruction AGE=0; avant que la boucle ne débute ?*

2.5 La gestion des valeurs manquantes

Dans la section 1.6, nous avons indiqué que vous deviez noter les valeurs manquantes dans vos données au moyen d'un '.', quel que soit le type de la variable – numérique ou caractère. SAS vous donne la possibilité, d'une part, de gérer plus finement ces valeurs manquantes et d'autre part, de noter la présence d'une valeur manquante dans vos données même si celle-ci n'est pas repérée au moyen du '.' habituel.

2.5.1 L'instruction globale MISSING

Imaginons le cas suivant : vous disposez d'un fichier de données qui reprend les résultats d'une enquête effectuée auprès de consommateurs sur les caractéristiques de leur abonnement auprès des opérateurs de téléphonie mobile. La première question posée est : « Quel est le nom de votre opérateur de téléphonie mobile ? »

Certains consommateurs n'ont pas répondu parce qu'ils ne le souhaitaient pas ; d'autres, parce qu'ils n'étaient pas concernés par la question (« Je n'ai pas de téléphone portable »). Dans votre fichier, deux caractères représentent les valeurs manquantes : '.' pour le premier cas et 'X' pour les personnes ayant déclaré ne pas avoir de téléphone portable. Nous construisons une table qui reprend une partie de ces données dans le programme 2.29.

Programme 2.29

```
DATA portable;
   INPUT ope X1 $;
   CARDS;
1 abc
X def
2 ghi
1 jkl
. mno
2 pqr
```

Résultat 2.11

Obs	ope	X1
1	1	abc
2	.	def
3	2	ghi
4	1	jkl
5	.	mno
6	2	pqr

La variable OPE est une variable numérique et la modalité X, *a priori* non numérique, est refusée lors de la création de la table. Le message habituel apparaît dans le JOURNAL :

```
OTE: Données incorrectes pour ope à la ligne 713 1-1.
EGLE : ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
13     X def
pe=. X1=def _ERROR_=1 _N_=2
```

Votre X a été transformé en valeur manquante, mais vous avez tout de même perdu une information : maintenant, vous ne pouvez plus donner la raison de cette non-réponse. Vous pouvez gérer ce genre de problème au moyen de l'instruction globale MISSING, qui permet de créer des valeurs manquantes spéciales. Le programme 2.30 intègre cette instruction globale.

ogramme 2.30

```
SSING X;
TA portable;
 INPUT ope X1 $;
 CARDS;
abc
def
ghi
jkl
mno
pqr

SING;
```

Résultat 2.12

Obs	ope	X1
1	1	abc
2	X	def
3	2	ghi
4	1	jkl
5	.	mno
6	2	pqr

Au terme du programme 2.30, nous revenons au fonctionnement par défaut de SAS *via* l'instruction MISSING;. Vous constatez (voir résultat 2.12) que le X est conservé et que la variable OPE est bien numérique.

L'instruction MISSING demande ici que X soit considéré comme symbole de valeur manquante. Dans cette instruction, vous pouvez indiquer soit une lettre (A-Z), soit l'underscore '_' (tiret bas), et affecter plusieurs symboles aux valeurs manquantes :

```
MISSING A B C;
```

Cette instruction indique à SAS qu'il doit considérer A, a, B, b, C et c – la casse est insensible – comme symboles possibles de valeurs manquantes. Une autre façon de faire comprendre (sans utiliser MISSING) qu'une lettre doit être saisie comme telle et être considérée ensuite comme valeur manquante d'une variable numérique consiste à la faire précéder d'un point.

Programme 2.31

```
DATA portable;
   INPUT x0 x1 $;
   CARDS;
1     abc
2     def
.A    ghi
.b    jkl
.ccc  mno
2     pqr
;
```

Résultat 2.13

Obs	x0	x1
1	1	abc
2	2	def
3	A	ghi
4	B	jkl
5	.	mno
6	2	pqr

Vous remarquez de plus, dans le programme 2.31, que les valeurs manquantes spéciales sont toujours affichées en majuscules ; par ailleurs, vous devez impérativement utiliser une seule lettre par valeur manquante spéciale.

2.5.2 Les valeurs manquantes non notées par un quelconque caractère

Lorsque les données sont formatées en colonnes, vous n'êtes pas obligé de noter les valeurs manquantes avec un point ou un autre caractère. En effet, si vous demandez à SAS de rechercher entre telle et telle colonne et qu'il ne trouve rien, il conclura que la modalité est manquante.

En revanche, si vos données sont de type LIST INPUT, l'absence de symbolisation des valeurs manquantes peut devenir un problème. Étudions cela, et la manière dont SAS peut gérer ce type de problème, à partir des données ci-après (contenues dans le fichier MISSING.TXT).

Données

```
a;b;c;1;2;3
d;e;f;4;5;6
g;h;i;7;8;9
k;l;m;10;11;12
```

Programme 2.32

```
DATA test;
   INFILE 'C:\intro_SAS\fichiers\missing1.txt' DLM=';';
   INPUT x1 $ x2 $ x3 $ n1 n2 n3;
RUN;
```

La table TEST est créée sans problème au moyen du programme 2.32. Mais si vos données ont la forme suivante (fichier MISSING2.TXT) et que vous exécutiez le même programme, attendez-vous à des surprises…

nnées

```
;c;1;2;3
f;4;5;6
i;7;8;9
;;10;;12
```

Résultat 2.14

Obs	x1	x2	x3	n1	n2	n3
1	a	b	c	1	2	3
2	e	f	4	5	6	.

Dans le fichier MISSING2.TXT apparaissent des valeurs manquantes non saisies par ';' : le résultat que vous obtenez est directement lié à cette caractéristique de vos données. Si vous voulez les utiliser correctement pour construire une table à l'image de vos données, et sans avoir à intervenir dans le fichier TXT, vous pouvez employer l'option DSD :

gramme 2.33

```
a test;
NFILE 'C:\intro_SAS\fichiers\missing2.txt' DLM=';' DSD;
NPUT x1 $ x2 $ x3 $ n1 n2 n3;
```

L'option DSD indique à SAS qu'il doit considérer deux séparateurs de champs consécutifs comme symbolisant une valeur manquante. Dans le programme 2.33, en utilisant les options DSD et DLM=, nous indiquons ';' comme délimiteur : deux points-virgules qui se suivent sont considérés comme le signe d'une valeur manquante. Si vous ne spécifiez pas de valeur à l'option DLM=, DSD considère que le séparateur de champs est la virgule. Deux virgules consécutives seront donc considérées comme marque d'une valeur manquante.

Vous pouvez aussi recourir à l'option DSD lorsque votre séparateur est l'espace.

ramme 2.34

```
 test;
NFILE CARDS DLM=' ' DSD;
NPUT x1 x2 x3;
S;
3
```

Programme 2.35

```
DATA test;
   INFILE CARDS DLM=' ' DSD;
   INPUT x1 x2 x3;
CARDS;
1 2 3
4   6
7 8 9
;RUN;
```

Dans le programme 2.34, puisque deux séparateurs de champs définis par DLM= se suivent, la variable X2 pour la seconde observation sera déclarée manquante. En revanche, ce type de programmation peut devenir dangereux si vous l'appliquez aux données reproduites dans le programme 2.35. Vous noterez en effet ici qu'entre 4 et 6, il y a trois espaces : SAS va exécuter le programme comme vous le lui avez ordonné et déclarera X3 comme valeur manquante pour le second individu. Maintenant, dans le cas présent, vous n'aurez aucune difficulté à gérer ce problème de valeur manquante non marqué par un signe quelconque puisque vos données sont présentées en colonnes.

2.6 Les INFORMAT

Le signe $ que vous devez ajouter à la suite du nom d'une variable caractère est un INFORMAT. Il a pour objet, comme tous les INFORMAT, de dire à SAS comment traiter une information contenue dans un champ pour la transformer en une modalité lisible par SAS. Il existe beaucoup d'INFORMAT et l'objet de cette section est de comprendre leur fonctionnement, leur utilisation et d'évoquer les plus courants[1].

2.6.1 Les INFORMAT des variables caractères

En ce qui concerne les variables caractères, il existe principalement deux INFORMAT : $w. et $CHARw. Le programme 2.36 illustre le fonctionnement de ces INFORMAT.

Programme 2.36

```
DATA clint4;
   INPUT titre_us $CHAR21. @23 titre_fra $21. @46 acteur $13.;
CARDS;
          Changeling L'échange                 Angelina Jolie
Letters from Iwo Jima Lettres d'Iwo Jima        Ken Watanabe
 Flags of Our Fathers Memoires de nos pères    Ryan Phillippe
;---+----1----+----2----+----3----+----4----+----5----+----6
```

Vous obtenez le résultat 2.15.

Résultat 2.15

Obs	titre_us	titre_fra	acteur
1	Changeling	L'échange	Angelina Joli
2	Letters from Iwo Jima	Lettres d'Iwo Jima	Ken Watanab
3	Flags of Our Fathers	Memoires de nos pères	Ryan Phillipp

Un INFORMAT doit être présenté immédiatement après le nom de la variable qu'il va contribuer à créer. Avec une variable caractère, le nom d'un INFORMAT se termine toujours par un '.'[2]. **La partie W de l'INFORMAT indique ici le nombre de caractères qu'il faut lire et enregistrer dans la variable.** Notre instruction INPUT demande donc que l'on lise **exactement** 21 caractères pour déterminer la modalité de TITRE_US. Il faut ensuite se rendre à la colonne 23 (@23) pour saisir, à partir de cette 23e colonne, les 21 caractères suivants comme modalité de la variable TITRE_FRA. Puis on passe en colonne 46 et l'on saisit à partir de cette colonne les 13 caractères qui suivent. Cette instruction INPUT est exactement équivalente à celle-ci :

```
INPUT titre_us $ 1-21 titre_fra $ 23-43 acteur $ 46-58;
```

Puisque vous spécifiez les colonnes à lire, vous n'avez plus besoin des pointeurs @X. Attention cependant : les deux instructions INPUT traitées ici sont parfaitement équivalentes mais vous ne pouvez pas indiquer pour une même variable, un INFORMAT et

1. Voir l'aide SAS, entrée « INFORMATS / categories of ».
2. Cela permet à SAS de faire la différence entre nom de variable et INFORMAT.

les colonnes dans lesquelles il doit lire une modalité – à moins, mais **c'est parfaitement inutile, voire dangereux**, d'indiquer par exemple, pour TITRE_US :

```
INPUT titre_us $CHAR21. 1-21 ...
```

À la lecture du résultat 2.15, vous remarquez que SAS fait exactement ce que vous lui avez demandé. Pour enregistrer la variable ACTEUR, vous avez indiqué la lecture de 13 caractères : il a lu 13 caractères, ce qui n'était pas suffisant, et a « coupé », oubliant ainsi la dernière lettre de chacun des noms de famille. Notez aussi que la présence d'espaces au sein des modalités n'arrête pas la lecture des W caractères que vous lui avez indiqués.

Enfin, cet exemple permet de voir la différence entre ces deux INFORMAT :

- `$CHARw.` saisit une chaîne de caractères et ne lui fait subir aucun traitement ;
- `$w.` élimine les espaces qui peuvent apparaître en début de modalité (alignement à gauche).

Un INFORMAT a donc la capacité de transformer le contenu d'un champ tel qu'il peut apparaître dans votre fichier en quelque chose de peut-être différent dans votre table finale. Vous pouvez ainsi comprendre les INFORMAT comme des **routines de transformation** des données présentes dans les champs en données SAS.

Exercice 2.12 : *Transformez le fichier LEADER_CCCP.TXT en table SAS. Il contient les données présentées ci-contre :*

Sans modifier quoi que ce soit dans cette liste, constituez une table SAS de deux variables (prénom et nom). Ne passez pas trop de temps sur cet exercice, car vous ne disposez pas des outils nécessaires à son achèvement. Tentez cependant de le résoudre afin de comprendre pourquoi cela ne fonctionne pas. Voyez dans un premier temps si vous avez besoin d'un autre INFORMAT que le $ que nous utilisons depuis le début.

```
----+----1----+----2
Vladimir-Ilych Lenin
Joseph Stalin
Georgy Malenkov
Nikita Khrushchev
Leonid Brezhnev
Yuri Andropov
Konstantin Chernenko
Mikhail Gorbachev
```

L'exercice 2.12 est important parce qu'il va vous permettre de comprendre l'INFORMAT $ et la raison pour laquelle il vous faudra très souvent utiliser des INFORMAT particuliers pour saisir les modalités des variables caractères.

Si vous avez exécuté le programme 2.37, vous avez obtenu le résultat 2.16.

Programme 2.37

```
DATA CCCP;
  INFILE 'C:\intro_SAS\fichiers\leader_cccp.txt';
  INPUT prenom $ nom $;
RUN;
```

Résultat 2.16

Obs	prenom	nom
1	Vladimir	Lenin
2	Joseph	Stalin
3	Georgy	Malenkov
4	Nikita	Khrushch
5	Leonid	Brezhnev
6	Yuri	Andropov
7	Konstant	Chernenk
8	Mikhail	Gorbache

Au moyen de cet exemple, vous comprenez que l'INFORMAT $ dispose des caractéristiques suivantes :

- Vous ne pouvez pas saisir plus de 8 caractères avec l'INFORMAT $.
- Après avoir lu 8 caractères, SAS passe les caractères qui suivent jusqu'à ce qu'il rencontre un séparateur de champs. Il commence alors l'enregistrement de la variable suivante[1].
- La fin de l'enregistrement est comprise comme un séparateur de champs : si un champ a moins de 8 caractères (LENIN), SAS ne cherche pas de champ de 8 caractères dans l'enregistrement suivant.

Attention, $8. et $ ne sont pas équivalents ! Dans le programme 2.37, remplacez $ par $8. : vous obtiendrez le résultat 2.17. Si vous utilisez ensuite $14. pour PRENOM (le prénom le plus long compte 14 caractères) et $10. pour NOM (longueur du nom le plus long), vous obtiendrez le résultat 2.18.

Résultat 2.17

Obs	prenom	nom
1	Vladimir	-Ilych L
2	Joseph S	Georgy M
3	Nikita K	hrushche
4	Leonid B	rezhnev
5	Yuri And	Konstant
6	Mikhail	Gorbache

Résultat 2.18

Obs	prenom	nom
1	Vladimir-Ilych	Joseph Sta
2	Georgy Malenko	Nikita Khr
3	Leonid Brezhne	Yuri Andro
4	Konstantin Che	Mikhail Go

Exercice 2.13 : *Expliquez pourquoi vous n'obtenez que 6 observations dans le résultat 2.17 et 4 observations dans le résultat 2.18.*

Lorsque vous utilisez un INFORMAT $w., comme indiqué plus haut, vous saisissez exactement W caractères. $ est donc différent puisqu'il va saisir **jusqu'à 8 caractères** et que la saisie cessera si SAS rencontre un séparateur de champs.

Vous avez donc besoin, pour traiter correctement le fichier LEADER_CCCP.TXT, d'un INFORMAT pouvant saisir jusqu'à W caractères (W=14 pour PRENOM et 10 pour NOM), mais qui peut cesser la saisie lorsqu'il rencontre un séparateur de champs : il s'agit de l'INFORMAT :$w.[2].

Si votre instruction INPUT devient :

```
INPUT prenom :$14. nom :$10.;
```

Vous obtiendrez le résultat 2.19. Si votre instruction INPUT est la suivante :

```
INPUT prenom :$10. nom :$10.;
```

Vous obtiendrez le résultat 2.20.

1. Autrement dit : avec l'INFORMAT $, SAS ne conserve d'un champ délimité par des séparateurs de champs que les 8 premiers caractères.
2. Il est possible d'ajouter un espace entre ':' et l'INFORMAT – c'est d'ailleurs ainsi qu'est présenté le modifieur ':' dans la documentation SAS. Nous préférons retirer cet espace pour pouvoir évoquer les INFORMAT précédés d'un ':' de la manière suivante :INFORMAT.

Résultat 2.19

Obs	prenom	nom
1	Vladimir-Ilych	Lenin
2	Joseph	Stalin
3	Georgy	Malenkov
4	Nikita	Khrushchev
5	Leonid	Brezhnev
6	Yuri	Andropov
7	Konstantin	Chernenko
8	Mikhail	Gorbachev

Résultat 2.20

Obs	prenom	nom
1	Vladimir-I	Lenin
2	Joseph	Stalin
3	Georgy	Malenkov
4	Nikita	Khrushchev
5	Leonid	Brezhnev
6	Yuri	Andropov
7	Konstantin	Chernenko
8	Mikhail	Gorbachev

Ainsi, lorsque vous ajoutez ':' avant le nom de votre INFORMAT, :$w. est capable :

- d'enregistrer **jusqu'à** W caractères ;
- de cesser la saisie si un séparateur de champs apparaît ;
- de cesser la saisie s'il arrive en fin de ligne ;
- si le champ comprend plus de caractères que W, d'aller jusqu'au prochain séparateur de champs pour passer à la saisie de la modalité de la variable suivante.

L'INFORMAT $ est donc parfaitement équivalent à un INFORMAT :$8.

Le tiret entre Vladimir et Ilych n'a aucune justification dans la grammaire russe, il est simplement présent pour vous simplifier la tâche. S'il n'y avait pas ce tiret, comment constitueriez-vous votre table SAS[1] ? Pour résoudre ce problème, vous pouvez introduire une petite modification dans le fichier tel qu'il apparaît plus haut, en plaçant par exemple deux espaces entre le prénom et le nom (voir section 2.2.2). Votre ligne INPUT deviendra :

```
PUT prenom & $14. nom :$10.;
```

Vous n'avez plus besoin de ':' pour l'INFORMAT de PRENOM puisqu'en utilisant &, vous demandez que cesse la saisie de la variable quand deux séparateurs de champs consécutifs apparaissent. Mais vous avez toujours besoin du deux-points pour NOM, puisque la saisie doit cesser lorsque SAS arrive à la fin de l'enregistrement.

Exercice 2.14 : *Quelle option d'INFILE devez-vous maintenant utiliser pour vous passer du ':' de l'INFORMAT de NOM ?*

Vous pouviez aussi choisir un autre séparateur de champs et utiliser l'option DLM= d'INFILE. Si votre séparateur de champs est maintenant égal à #, pour construire correctement vos tables, INFILE et INPUT deviennent :

```
ILE 'C:\intro_SAS\fichiers\leader_cccp.txt' DLM='#';
UT prenom :$14. nom :$10.;
```

Mais attention, n'utilisez pas de tabulation comme séparateur de champs (voir section 2.2.1).

'ans ce cas, les champs ne sont pas présentés en colonnes, mais il existe des espaces au sein des modalités des variables ıractères.

2.6.2 Autres INFORMAT caractères

Il existe d'autres INFORMAT permettant la création de variables caractères. Pour une liste complète, consultez l'aide SAS[1]. Le tableau 2.1 vous en présente quelques-uns.

Tableau 2.1 : INFORMAT des variables caractères

Donnée	INFORMAT appliqué	Modalité dans SAS
"bourvil"	$QUOTE9.	bourvil
'fernandel'	$QUOTE11.	fernandel
SAS 9'2"	$QUOTE9.	SAS 9'2
Bourvil	$UPCASE3.	BOU
Bourvil	$UPCASE7.	BOURVIL
01010010	$BINARY.	R
01101110	$BINARY.	n

Exercice 2.15 : *Vous disposez à ce stade de l'ensemble des connaissances nécessaires au traitement du fichier BLUENOTE.TXT (voir exercice 2.5). Créez deux tables SAS à l'aide de ce fichier. La première table SAS doit présenter de la manière suivante les 100 références du catalogue BLUE NOTE citées dans ce fichier :*

Obs	ref	titre
1	blp-4201	Stanley Turrentine - Joyride
2	blp-4202	Grant Green - I Want To Hold Your Hand

La seconde table devra présenter ainsi les informations contenues dans ce fichier :

Obs	ref	artiste	album
1	BLP 4201	Stanley Turrentine	Joyride
2	BLP 4202	Grant Green	I Want To Hold Your Hand

L'exercice est difficile… Regardez à présent les tables que vous avez réussi à créer (avec un peu de persévérance). Vous voyez apparaître quelques «scories» : des '"' ou des 'quot;', des '/a>'… Nous reviendrons sur ces tables dans un nouvel exercice, afin de corriger cela…

2.6.3 Longueur des variables caractères

Il est extrêmement important de comprendre les incidences de la partie W des INFORMAT. Avec un INFORMAT sur variable caractère, le W indique deux choses :

- le nombre de caractères à lire dans le fichier, information nécessaire si votre modalité compte plus de 8 caractères ou si un espace intervient dans la modalité ;
- le nombre d'octets à utiliser pour enregistrer la modalité dans la table SAS, et donc **le nombre maximal de caractères** qui pourront être enregistrés pour cette variable[2].

Nous avons ainsi vu que $ ne pouvait pas saisir plus de 8 caractères à la fois lors de la création d'une table. Par la suite, lorsque vous modifierez votre table, vous ne pourrez pas non plus modifier la modalité de votre variable pour lui associer une chaîne de plus de 8 caractères. Nous expliquerons ce point dans la section 3.1.

1. Entrée «INFORMATS / categories of».
2. En effet, pour enregistrer un caractère, il faut un octet.

2.6.4 Les INFORMAT des variables numériques

Vous n'avez pas besoin de spécifier un INFORMAT si le champ que vous avez à transformer en une modalité de variable SAS est de la forme :

23 23.2 –23 00023 2.3E1 2.3E01 230E-1

Les données numériques ont cependant parfois des formes qui ne conviennent pas à SAS. Nous avons ainsi vu que le séparateur décimal interne à SAS était le point. Si l'on essaie d'entrer des données avec des virgules, SAS interprète ce type de données comme un caractère et crée des valeurs manquantes dans la table. Les INFORMAT numériques permettent de gérer ce type de difficulté.

a. Principes

Les INFORMAT des variables numériques fonctionnent quelque peu différemment des INFORMAT caractères, car le statut du W n'est pas le même.

En ce qui concerne les variables numériques, le W ne spécifie pas le nombre d'octets à utiliser pour enregistrer la modalité dans la table SAS[1]. Il ne sert qu'à indiquer le nombre de caractères à lire dans le champ. Ensuite, contrairement aux variables caractères, SAS ne limite pas sa lecture à 8 caractères[2]. Par conséquent, sauf cas particuliers, vous n'aurez pas besoin de spécifier le W des INFORMAT numériques, et la plupart de vos INFORMAT seront de la forme :INFORMAT.

Vous devez prendre garde à la spécification de votre W uniquement :

- Lorsque le séparateur de champs est l'espace (ou la tabulation) et que des espaces apparaissent au sein du champ que vous souhaitez transformer en une modalité SAS[3].

  Exemple : 1 000, 1er janvier 2011

- S'il n'y a pas de séparateur de champs et que votre donnée numérique présente toujours le même nombre de caractères.

  Exemple : 1000abc 0043def (pour des modalités de 1000 et de 43 pour la variable numérique.)

Vous pouvez utiliser des :INFORMAT sans préciser W (ou en donnant une valeur quelconque à W) :

- S'il n'y a pas d'espace dans les champs.
- Si des espaces apparaissent dans les champs et si le séparateur de champs n'est ni l'espace, ni la tabulation.

Le programme 2.38 illustre ce fonctionnement et précise certains points.

Programme 2.38

```
DATA illustration;
   INFORMAT x1 PERCENT25.;
   INPUT X1 (X2 X3) (:EUROX1.) X4 :NUMX.;
CARDS;
12.25% €1 €2.000,00  1,1
4.5% €1.000,00 €2,0  11,1
3% €1,00 €4.000.000  1111,1
;
```

Vous pouvez préciser l'INFORMAT d'une variable en dehors de l'instruction INPUT *via* une instruction INFORMAT. Dans ce cas, les INFORMAT cités seront de type :INFORMAT. Vous remarquez en effet que 25 caractères ne sont pas lus pour peupler la variable X1 – la lecture de l'enregistrement pour X1 s'arrête au premier séparateur de champs.

Ce programme permet aussi d'imposer à plusieurs variables le même INFORMAT. Vous devez regrouper les variables entre parenthèses, de même que l'INFORMAT à appliquer. Vous pouvez donner n'importe quelle valeur à la partie W de l'INFORMAT, et même ne rien préciser : cette partie W n'est en fait jamais considérée si vous utilisez ':'.

Exercice 2.16 : *w.d est l'INFORMAT numérique le plus basique (la partie D de l'INFORMAT est optionnelle et expliquée plus bas). Il est très peu utilisé mais afin de bien comprendre le fonctionnement des INFORMAT, nous le mobilisons dans le programme 2.39 (alors que les données ne nécessitent aucun INFORMAT).*

Programme 2.39

```
DATA test;
   INPUT x1 1. x2 2. x3 3. x4 4.;
   CARDS;
1 23 456 7890
2 34 567 8901
3 45 678 9012
;
```

Résultat 2.21

Obs	x1	x2	x3	x4
1	1	2	.	.
2	2	3	.	.
3	3	4	.	.

Le résultat 2.21 présente la table construite à l'aide de ce programme. Expliquez ce résultat. En utilisant ***uniquement des INFORMAT sans ':' et sans prendre en compte le fait que les données sont formatées en colonnes****, construisez la table correspondant à vos données. (Une relecture de la section 2.3.2 vous aidera sûrement pour cet exercice…)*

b. Quelques INFORMAT sur variables numériques

Les nouvelles versions de SAS sont toujours l'occasion d'introduire de nouveaux INFORMAT. Pour connaître la liste complète des INFORMAT à disposition pour votre version de SAS, consultez l'aide SAS[1]. Nous ne présenterons ici que les INFORMAT les plus courants, avec la notation utilisée par l'aide SAS, à savoir INFORMATw.d[2]. Vous utiliserez le plus souvent ces INFORMAT sous leur forme :INFORMAT.

1. Entrée « INFORMATS / categories of ».
2. Si un INFORMAT ou un FORMAT utilise la possibilité offerte par le d, il n'est jamais suivi d'un point. Le point que vous observez ici termine la phrase.

·us pouvez tout de même fixer le nombre d'octets utilisés pour stocker une modalité numérique ; mais cela n'est pas ssible avec l'instruction INPUT – les données numériques sont systématiquement codées sur 8 octets.

ıns une étape DATA incluant un CARDS, SAS accepte les chiffres composés au maximum de 308 caractères.

: cas est très improbable. En effet, pour que vous puissiez construire une table qui comprendrait plusieurs variables, si space intervient à la fois au sein de la modalité et comme séparateur de champs, il faudrait que votre variable numérique ésente toujours le même nombre de caractères ou que vos données soient présentées en colonnes.

Généralement, un INFORMATw.d demande la lecture de W caractères exactement, et la multiplication par 10^{-D} de la donnée lue si celle-ci est entière. **Attention, si la donnée est décimale, la multiplication par 10^{-D} ne sera pas effectuée.**

Lorsque vous recourez à un :INFORMATw.d, SAS prend en compte la partie D de l'INFORMAT, mais pas la partie W. Si vous souhaitez utiliser uniquement D, vous pouvez indiquer n'importe quelle valeur pour W ou bien ne pas le préciser.

NUMXw.d - Permet de lire les données avec le séparateur décimal (virgule).

:NUMX. va transformer 896,48 (donnée) en 896.48 (donnée SAS).

Exercice 2.17 : *Sur la base du fichier MARDI.TXT que vous ouvrirez auparavant au moyen d'un éditeur de texte quelconque, créez une table SAS.*

PERCENTw.d - Supprime les points, les blancs, les signes $, les tirets, les parenthèses droites et transforme les parenthèses gauches en signe – (s'il n'y a pas de signe – entre les parenthèses).

Ainsi :

Donnée	INFORMAT	Modalité SAS
1%	:PERCENT.	0.01
1.6%	:PERCENT.	0.016
(1%)	:PERCENT.	–0.01
(–1%)	:PERCENT.	–0.01
1 %	PERCENT3.	0.01
–1%$	:PERCENT.	–0.01
1	:PERCENT.	1
1)	:PERCENT.	1

En revanche, si le signe % n'apparaît pas, la donnée n'est pas divisée par 100. Pour la donnée 1 %, vous devrez préciser le nombre de caractères à lire étant donné qu'un espace apparaît entre le 1 et le signe %.

Exercice 2.18 : *Créez deux tables SAS au moyen des fichiers TAUX.TXT et TAUXV.TXT. Vous aurez quelques difficultés avec le second fichier… Recherchez dans l'aide des informations sur l'INFORMAT NLPCTw.d.*

COMMAw.d - Répond à un problème spécifique aux Américains. Lorsque ceux-ci écrivent 1 000 000 de dollars, cela donne $1,000,000.

:COMMA. transforme $1,000,000 (donnée) en 1000000 (donnée SAS).

COMMAw.d supprime les signes $, les blancs, les pourcentages, les virgules, les tirets, les parenthèses droites et transforme les parenthèses gauches en signes –. La transformation par 10^{-D} est aussi possible.

Exercice 2.19 : *Appliquez l'INFORMAT COMMAw.d aux données suivantes que vous aurez, au préalable, replacées dans une seule colonne.*

```
1,000,000  $1,000,000  1 000 000  1-000-000  1 000 000%
(1,000,000)  (-1,000.00)  €1.000.000
```

Retirez les valeurs présentant des espaces pour vérifier que :COMMA. traite les autres valeurs (sauf la dernière…).

COMMAXw.d - Réalise les mêmes transformations sauf qu'elles s'appliquent aux données du type 1.000.000 (notation à l'européenne de 1 000 000).

EUROw.d - Retire les virgules et les signes € (le signe E pour l'euro n'est plus reconnu par SAS depuis la version 9.1) qui peuvent apparaître au format « à l'américaine » (avant la somme).

:EURO. transforme €1,000,000 (donnée) en 1000000 (donnée SAS).

EUROXw.d - Réalise les mêmes transformations sur les sommes saisies à l'européenne.

:EUROX. transforme €1.000.000 (donnée) en 1000000 (donnée SAS).

Exercice 2.20 : *Dans le fichier PABLO, vous trouverez un descriptif du catalogue des disques PABLO (extrait) – label de jazz des années 1970 créé par Norman Granz, créateur du label Verve.*

Les disques de ce label ont fait l'objet de rééditions (collection OJC) en disques trente-trois tours vinyle et étaient encore récemment distribués par deux sociétés: Fantasy aux États-Unis et ZYX en Allemagne. Dans ce fichier sont indiqués, dans l'ordre, la référence PABLO du disque, l'artiste, le titre de l'album, si Fantasy le vend en format vinyle (fantasy/no), la référence OJC en cas de vente par Fantasy, le prix Fantasy en dollars, si ZYX le vend en format vinyle (ZYX/no) et le prix ZYX en euros.

Créez une table SAS à partir des données présentées et, surtout, étudiez la structure du fichier PABLO au moyen d'un éditeur de texte quelconque avant de lancer quoi que ce soit.

Exercice 2.21 : *Construisez une table SAS en utilisant le fichier PABLO2.TXT (catalogue complet – la structure du fichier diffère très nettement).*

Réservez 48 caractères pour l'artiste (la variable artiste peut prendre jusqu'à 95 caractères : vous allez donc la tronquer), 40 caractères pour le titre de l'album (qui peut en compter jusqu'à 70). La structure des données est la suivante :

```
----+----1----+----2----+----3----+----4----+
2312131 Count Basie & His Orchestra
Warm Breeze  fantasy
OJC-994 $9.98 ZYX €11,49
```

2.6.5 Des variables numériques particulières : les dates et heures

Les dates et heures sont des variables numériques. Si une variable reprenant une date apparaît par exemple dans votre fichier sous la forme '01JAN2011', vous devez, par le biais d'un INFORMAT, faire comprendre à SAS, lors de la création de votre table, qu'il s'agit bien d'une date. Sinon, en enregistrant cette modalité dans une variable caractère, vous ne pourrez pas, par exemple, regrouper les observations relatives à un même mois, trier vos données par ancienneté ou calculer un temps entre deux observations consécutives.

a. Qu'est-ce qu'une date ?

SAS stocke une date sous la forme d'une variable numérique qui indique le nombre de jours séparant la date en question du 01/01/1960 : un chiffre négatif indiquera donc une date antérieure au 1er janvier 1960. SAS gérera n'importe quelle date entre 1582 (année d'imposition du calendrier grégorien) et 20 000.

La gestion des dates par SAS est pour partie automatisée.

Programme 2.40

```
DATA _NULL_;
   date="01JAN2011"d;
   ecart="01JAN2012"d-"01JAN2011"d;
   PUT date= ecart=;
RUN;
```

L'instruction PUT (voir section 2.7.4) utilisée ici permet de voir dans la fenêtre JOURNAL le résultat suivant :

```
date=18628 ecart=365
```

C'est ici le seul résultat attendu du programme et il n'est pas réellement utile de créer une table SAS contenant les variables DATE et ECART. L'instruction DATA _NULL_ est alors utilisée : elle permet de tirer profit de l'ensemble des outils de programmation de l'étape DATA sans pour autant créer une table.

Normalement, lorsque vous attribuez une modalité à une variable caractère, vous présentez cette modalité entre 'quotes' (voir section 1.7). Ici, parce que nous indiquons juste après la quote fermante un « d », nous indiquons à SAS qu'une date est présentée entre quotes et qu'il doit interpréter cette date. Cette interprétation est visible dans le résultat proposé : il y a 18 628 jours entre le 1er janvier 2011 et le 1er janvier 1960 – l'année 2011 compte bien 365 jours.

Pour être interprétable, une date entre quotes doit être présentée dans un FORMAT DATE9. (01JAN2011) ou DATE7 (01JAN11)[1].

Pour créer une variable numérique date qui aura pour modalité un nombre de jours depuis le 1er janvier 1960, vous disposez aussi de la fonction MDY.

Programme 2.41

```
ATA test;
  INPUT jj mm aa;
  date=MDY(mm,jj,aa);
ARDS;
 1 1960
 1 1900
 12 2011
 02 2011
UN;
```

Résultat 2.22

Obs	jj	mm	aa	date
1	1	1	1960	0
2	1	1	1900	-21914
3	31	12	2011	18992
4	29	2	2011	.

La fonction MDY vous permet de recoder un ensemble de variables dans la forme que SAS comprend comme étant une date. Attention : SAS comprend les dates à l'américaine, c'est-à-dire mois/jours/année (MDY : *Month Day Year*).

La fonction MDY est « intelligente » : elle produit une valeur manquante si la date que vous essayez de lui faire comprendre n'existe pas (pas de 29 février en 2011).

La date peut être rendue plus parlante par l'application d'un FORMAT. Le FORMAT définit la forme que devra prendre la variable à l'affichage[2].

[1] Les mois doivent être saisis sur trois lettres – il s'agit impérativement des abréviations anglaises des mois, soit JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC. Attention au YEARCUTOFF si vous n'indiquez que deux chiffres dans votre date (voir section 2.6.5.c).

[2] Nous traiterons des FORMAT plus en détail dans la section 5.3.

Programme 2.42

```
DATA test;
   SET test;
   FORMAT date YYMMDD10.;
RUN;

PROC PRINT;
;RUN;
```

Résultat 2.23

Obs	jj	mm	aa	date
1	1	1	1960	1960-01-01
2	1	1	1900	1900-01-01
3	31	12	2011	2011-12-31
4	29	2	2011	.

b. L'utilisation des INFORMAT de dates

Lors de la phase de création de votre table, vous pouvez aussi demander, *via* des INFORMAT de dates (voir les plus courants dans le tableau 2.2), l'interprétation de chaînes de caractères contenues dans votre fichier et représentant des dates.

Imaginons que dans le fichier que vous devez transformer en table SAS, la date du 1er décembre 1960 apparaisse sous la forme suivante : 01/12/1960.

Programme 2.43

```
DATA date3;
   INPUT date :DDMMYY.;
CARDS;
01/12/1960
02/12/1960
13/12/1960
;RUN;
```

Assurez-vous de spécifier le bon INFORMAT : ici, les modalités sont bien 1er décembre 1960, 2 décembre 1960, 13 décembre 1960. Dans le meilleur des cas, si vous vous trompez d'INFORMAT, SAS ne comprendra pas et votre modalité sera valeur manquante. Il demeure possible qu'il interprète malgré tout votre champ. Si au lieu de :DDMMYY., vous utilisez :MMDDYY., SAS comprendra vos dates « à l'américaine » et les codifiera en interne comme étant les 12 janvier 1960, 12 février 1960. Pour la dernière modalité, vous aurez une valeur manquante.

Si vous devez et pouvez spécifier le W, il se peut que les INFORMAT incluant les W ne comprennent pas les dates écrites dans un format voisin. DDMMYY10. peut ainsi transformer 1/1/1960, 01/01/1960, 01-01-1960, 01011960, 01/01/60, 010160 ou 01 01 1960 en une date SAS. DDMMYY6. ne comprendra pas 01/01/1960. En revanche, si vous pouvez ne pas préciser le W, l'INFORMAT :DDMMYY. comprendra toutes les formes possibles dans lesquelles il n'y a pas d'espace.

Programme 2.44

```
DATA test_d;
   INPUT x :DDMMYY. @@;
CARDS;
01102012 011012 01-10-2012 01-10-12 01/10/2012 01/10/12 01.10.2012 01.10.12 01:10:2012 01:10:12
1102012 011012 1-10-2012 1-10-12 1/10/2012 1/10/12 1.10.2012 1.10.12 1:10:2012 1:10:12
;RUN;
```

Tableau 2.2 : Liste des INFORMAT de dates les plus courants (1er octobre 2012)

Dates dans votre fichier	INFORMAT dédié	Forme générale
1OCT12	DATE7.	:DATE.
1OCT12	DATE.	
01OCT12	DATE7.	
01OCT2012	DATE9.	
01-OCT-12	DATE9.	
011012	DDMMYY6.	:DDMMYY.
01 10 12	DDMMYY8.	
01/10/12	DDMMYY8.	
01/10/2012	DDMMYY10.	
12274	JULIAN.	:JULIAN.
2012274	JULIAN7.	
100112	MMDDYY.	:MMDDYY.
100112	MMDDYY6.	
10/01/12	MMDDYY8.	
10/01/2012	MMDDYY10.	
Oct12	MONYY.	:MONYY.
Oct2012	MONYY7.	
121001	YYMMDD6.	:YYMMDD.
121001	YYMMDD.	
12-10-01	YYMMDD8.	
12 10 01	YYMMDD8.	
2012-10-01	YYMMDD10.	
12Q4	YYQ.	:YYQ.
2012Q4	YYQ6.	

Attention : ce tableau étant loin d'être complet, consultez l'aide SAS[1] pour connaître les INFORMAT de dates disponibles pour votre version de SAS. Un INFORMAT supplémentaire, à savoir ANYDTDTEw, mérite d'ailleurs une mention spéciale.

Cet INFORMAT peut être utilisé lorsque les champs prennent plusieurs formes pour créer une même variable. Le tableau 2.3 propose des exemples d'application sur divers champs. Il montre aussi quelques limites à la capacité d'interprétation des champs avec ANYDTDTEw.

. Entrée « INFORMATS / categories of ».

Tableau 2.3

Champ	Modalité dans la table SAS avec ANYDTDTEw.	Interprétation de cette date	Commentaire (INFORMAT alternatif)
02JAN09	17899	02/01/2009	OK (DATE7.)
02JAN2009	17899	02/01/2009	OK (DATE9.)
020109	17899	02/01/2009	OK (DDMMYY6.)
02012009	17899	02/01/2009	OK (DDMMYY8.)
2009002	17899	02/01/2009	OK (JULIAN7.)
09002	17899	02/01/2009	OK (JULIAN5.)
January 2, 2009	17899	02/01/2009	OK
010209	17929	01/02/2009	Erreur : MMDDYY8. mal compris
01022009	17929	01/02/2009	Erreur : MMDDYY10. mal saisis
02-01-09 11:23	17899	02/01/2009	OK (DATETIME.)
02-01-09 06:53 pm	.	.	Erreur : MDYAMPM. mal compris - pb avec le PM
090201	15015	09/02/2001	YYDDMM8. mal compris
20090201	17929	01/02/2009	YYDDMM10. compris

Retenez ceci : ANYDTDTE est capable d'interpréter bon nombre de champs qui auraient normalement nécessité des INFORMAT comme DATEw., DDMMYYw. ou JULIANw[1]. Par contre, certains champs peuvent être ambigus. Ainsi, 02/01/09 peut aussi bien être interprété comme :

- le 2 janvier 2009 (DDMMYYw.) ;
- le 1er février 2009 (MMDDYYw.) ;
- le 1er septembre 2002 (YYDDMMw.) ;
- le 9 janvier 2002 (YYMMDDw.).

En cas d'ambiguïté, SAS considère la valeur donnée à l'option système DATESTYLE=. Pour un SAS « français », cette valeur est égale par défaut à DMY – ainsi, en cas d'ambiguïté, les champs seront toujours lus en considérant que le jour est présenté en premier, suivi du mois et enfin de l'année. Vous avez bien entendu la possibilité de donner une autre valeur à cette option, en fonction des données que vous aurez à mobiliser pour construire votre table SAS (voir l'aide SAS[2]).

Vous pouvez également inclure des dates saisies en français ou dans d'autres langues. Les langues comprises par SAS sont les suivantes :

Afrikaans* AFR	Finnish FIN	Macedonian* MAC	Slovenian* SLO
Catalan* CAT	French FRA	Norwegian NOR	Spanish ESP
Croatian* CRO	German DEU	Polish* POL	Swedish SVE
Czech* CSY	Hungarian* HUN	Portuguese PTG	Swiss_French FRS
Danish DAN	Italian ITA	Russian* RUS	Swiss_German DES
Dutch NLD			

* Ces langues ont été introduites avec SAS 9.2.

1. Pour plus de détails sur ANYDTDTEw., voir l'aide SAS.

Vous aurez pour cela recours aux INFORMAT suivants : EURDFDEw. (version internationale de DATEw.) et EURDFMYw. (version internationale de MONYYw.). Vous pouvez les utiliser dans la forme internationale en spécifiant une option de langue – ou simplement modifier le préfixe. Ainsi :

Programme 2.45[1]

```
DATA date4;
   INPUT date FRADFDE.;
CARDS;
01JAN60
01FEV60
01MAR60
01AVR60
01MAI60
01JUN60
01JUL60
01AOU60
01SEP60
01OCT60
01NOV60
01DEC60
;RUN;
```

```
OPTIONS DFLANG=french;
DATA date4;
   INPUT date EURDFDE.;
CARDS;
01JAN60
01FEV60
01MAR60
01AVR60
01MAI60
01JUN60
01JUL60
01AOU60
01SEP60
01OCT60
01NOV60
01DEC60
;RUN;
```

Supposons maintenant que votre date soit saisie de telle sorte que SAS ne la comprend pas, même en utilisant un de ses INFORMAT. Vous avez alors deux manières de procéder :

- Soit vous recodez les jours, mois et années (trois variables distinctes) pour ensuite employer la fonction MDY. Exemple : 1er juillet 2011.
- Soit vous créez votre propre INFORMAT (voir section 2.6.6).

Exercice 2.22 : *Vous trouverez, parmi les fichiers de l'archive que vous avez téléchargée sur www.sas-sr.com, un fichier EXO2XXX.TXT contenant 10 variables. Examinez-le attentivement. Construisez à partir de ce fichier une table SAS dans laquelle il ne doit y avoir aucune valeur manquante.*

c. L'option YEARCUTOFF=

Soit la date suivante qui apparaît dans un fichier TXT : 01/01/07. Quelle est exactement la date qui se cache derrière ce 01/01/07 : le 1er janvier 2007, 1907, 1807… ?

Par défaut, la date sera comprise (et recodée) comme étant le 1er janvier 2007. Dans le cas où 07 signifierait en fait 1907, vous devrez spécifier une nouvelle valeur à l'option YEARCUTOFF=.

```
OPTIONS YEARCUTOFF=yyyy;
```

Par défaut, le YEARCUTOFF est égal à 1920 : pour SAS, la date 01/01/10 correspond forcément au 1er janvier 2010, et la date 01/01/19 à 01/01/2019 ; mais il traduit le 01/01/20 en 01/01/1920. Si vous souhaitez que le 01/01/20 soit considéré par SAS comme le 01/01/2020, vous devez modifier le YEARCUTOFF en lui donnant une valeur supérieure à 1920. Cette option est inutile si vous entrez vos dates sous la forme 01/01/1789.

Exercice 2.23 : *Quel YEARCUTOFF faut-il spécifier pour que 14/07/89 soit traité par SAS comme étant le 14 juillet 1789 ?*

1. JUI n'est jamais une abréviation correcte – employez JUN pour juin et JUL pour juillet.

d. Une seconde présentation possible des dates : DATETIME

Vous trouverez assez souvent des dates saisies sous la forme suivante : 01JAN2011:01:23:45. Afin de saisir correctement ce champ dans une table SAS, employez l'INFORMAT DATETIMEw. Le champ sera alors recodé par SAS comme un nombre de secondes écoulées entre la date en question et le 1er janvier 1960, 00h00.

Dans vos programmes SAS, vous utiliserez le suffixe 'dt' pour faire appel à des dates ainsi codées.

Programme 2.46

```
DATA _NULL_;
   date="01JAN2011:00:00:00"dt;
   ecart="01JAN2011:01:23:45"dt-date;
   PUT date= ecart=;
RUN;
```

Vous obtenez dans la fenêtre JOURNAL le résultat suivant :

```
date=1609459200 ecart=5025
```

Afin que DATETIMEw. puisse la gérer, la partie date doit être présentée sous la forme DATE7. (01JAN11) ou DATE9. (01JAN2011). Vous pouvez utiliser le blanc ou n'importe quel caractère (sauf un chiffre) pour séparer la partie date de la partie heure. La partie heure doit être notée au format suivant :

```
hh:mm:ss.ss
```

Vous pouvez ne pas préciser ss.ss, mais les heures et minutes (hh:mm) doivent obligatoirement être spécifiées. AM (am) et PM (pm) peuvent suivre votre date, séparée ou non par un espace. Si votre date n'a pas exactement cette forme, il existe d'autres INFORMAT (YMDDTTMw.d, MDYAMPMw.d et les INFORMAT ISO8601 – voir l'aide SAS[1]), ainsi qu'une version internationale de DATETIMEw. (EURDFDTw.), qui fonctionne selon des principes identiques à ceux exposés pour les INFORMAT internationaux de dates.

Enfin, l'INFORMAT ANYDTDTMw. vous permet de produire des modalités de type DATETIME à partir de champs présentés sous diverses formes (voir l'aide SAS). Si seule la partie heure d'un champ de type DATETIME vous intéresse, vous pouvez employer l'INFORMAT ANYDTTMEw. (voir l'aide SAS).

e. Les INFORMAT d'heures

Le principe des variables d'heures est identique à celui des variables de dates. Les mesures du temps stockées par SAS correspondent au nombre de secondes qui séparent minuit de l'heure considérée. Ainsi, 23:59:59.9, soit un dixième de seconde avant minuit, sera enregistré par SAS sous la forme 86399.9 (0.9 + 59 + 59 * 60 + 23 * 60 * 60).

L'INFORMAT TIMEw. permet de gérer des champs présentés ainsi :

```
hh:mm:ss<.ss> <AM | PM>
```

1. Entrée « INFORMATS / categories of ».

Les parties centièmes de seconde et AM|PM sont optionnelles. Bien que cela ne soit pas mentionné dans l'aide SAS, vous n'êtes pas obligé de préciser les secondes. 12:34 sera compris comme 12 heures et 34 minutes (et aura 45240 pour modalité dans votre table SAS (12*60*60+34*60)).

Si vous avez des champs de la forme 2:30 pour 2 minutes et 30 secondes, vous ne pouvez pas recourir à l'INFORMAT TIMEw. car celui-ci interprétera 2:30 comme « deux heures et 30 minutes ». Vous utiliserez alors l'INFORMAT STIMERw. (voir l'aide SAS).

Vous avez aussi la possibilité de faire directement référence à des heures dans vos programmes – vous emploierez à cet effet le suffixe 't' :

gramme 2.47

```
_NULL_;
ieure="12:34"t;
ieure2="12:34:28"t;
ieure3="0:34:28.87 PM"t;
UT heure= heure2= heure3=;
;
```

Pour obtenir dans la fenêtre JOURNAL :

```
re=45240 heure2=45268 heure3=45268.87
```

2.6.6 Créons notre propre INFORMAT[1] !

Il vous arrivera très certainement d'avoir à traiter des données pour lesquelles vous aurez besoin d'un INFORMAT qui n'existe pas. SAS vous offre la possibilité de créer vous-même des INFORMAT grâce à la procédure PROC FORMAT.

Prenons un exemple. Votre entreprise a réalisé une enquête, sans vous consulter, et a saisi de la manière suivante les réponses aux questions posées : oui/non/ne sait pas. De plus, elle a sous-traité la saisie des réponses aux questionnaires papier à une entreprise qui a bâclé le travail : les réponses contenues dans votre fichier sont de la forme oui/OUI/Oui ; non/NON/Non ; ne sait pas/Ne Sait Pas/NE SAIT PAS. En revanche, votre responsable vous indique qu'ils avaient prévu qu'un problème pouvait se poser : ils ont fait mettre deux espaces entre chaque champ dans le fichier. Pour créer votre INFORMAT, vous avez ici le choix entre deux méthodes. La méthode simple et une méthode plus compliquée, mais qui vous permettra de vous sortir de cas beaucoup plus ardus que celui exposé ici.

Attention : cette section fait appel à des notions que nous n'avons pas évoquées jusqu'à présent (mais que nous verrons par la suite). Si vous lisez cet ouvrage pour la première fois, vous pouvez passer cette section et y revenir par la suite. Nous n'évoquerons pas ici la sauvegarde pour utilisation ultérieure des INFORMAT. Ce point sera plus particulièrement traité dans la section 5.3.5.c

a. Cas simples

Un cas est simple lorsque vous n'avez que quelques modalités possibles pour une variable. Ainsi, dans l'exemple présenté ici, nous avons neuf champs (réponses) possibles que nous allons transformer en trois modalités (1, 2 et 3). Dans le cas simple, vous pouvez passer directement par PROC FORMAT.

Programme 2.48

```
PROC FORMAT;
  INVALUE enq
        'OUI','Oui','oui'=1
        'NON','Non','non'=2
        'NE SAIT PAS', 'ne sait pas', 'Ne Sait Pas'=3;
RUN;
```

Le programme présenté ici construit un INFORMAT numérique (puisque les modalités prises par la variable réponse seront par la suite 1, 2 et 3). On pourrait construire un INFORMAT caractère de la même manière :

Programme 2.49

```
PROC FORMAT;
  INVALUE $enq
        'OUI','Oui','oui'='O'
        'NON','Non','non'='N'
        'NE SAIT PAS', 'ne sait pas', 'Ne Sait Pas'='NSP';
RUN;
```

Les remarques qui suivent sont valables que vous créiez un INFORMAT ou un FORMAT[1] :

- L'instruction de création d'un INFORMAT (FORMAT) est INVALUE (VALUE).
- Le nom de l'INFORMAT (FORMAT) ne doit pas prendre plus de 7 caractères, signe dollar compris, si vous disposez de SAS 8, ou plus de 31 caractères si vous avez la version 9.
- L'INFORMAT est caractère si les modalités, **à droite du signe égal**, sont précisées entre quotes (cas du programme 2.49)[2].
- Dans ce cas, le nom de votre INFORMAT (FORMAT) doit commencer par le signe $.
- Vous ne pouvez pas prendre pour nom celui d'un INFORMAT (FORMAT) existant.
- Votre INFORMAT/FORMAT ne peut se terminer par un chiffre.

Dans le cas des INFORMAT/FORMAT caractères, les modalités que vous allez associer à certains champs peuvent compter jusqu'à 32 767 caractères depuis SAS 9. Pour spécifier les champs à associer à une modalité, vous pouvez les saisir comme :

- Une valeur unique : 4 ou 'A' ou 'OUI'.
- Une liste de valeurs de même type séparées par des virgules : 4, 5, 12, 23 ou 'A', 'B', 'Z', 'OUI'.

1. Voir la section 5.3.5
2. Dans le cas des FORMAT, les modalités sont précisées à gauche du signe égal et les valeurs que vous voulez voir affichées sont à droite. Le FORMAT est caractère si les modalités sont précisées entre quotes.

- Un intervalle de valeurs caractères. Ainsi, pour toute lettre comprise entre M et Z : 'M'-'Z'.
- Un intervalle de valeurs numériques. Ainsi, pour toute valeur numérique comprise entre 1 et 50 : 1-50.

Pour spécifier des intervalles numériques, vous pouvez indiquer :

- low-13 : inférieur ou égal à 13.
- low-<13 : strictement inférieur à 13.
- 25-high : supérieur ou égal à 25.
- 25<-high : strictement supérieur à 25.
- Si vous spécifiez mal vos intervalles, s'ils se chevauchent comme dans 14-25 et 25-high, le second intervalle sera traité comme l'intervalle « strictement supérieur à 25 ».
- Le terme OTHER est possible pour associer une modalité à tout champ non évoqué par les intervalles. OTHER doit cependant intervenir en dernier.

Que se passe-t-il si vous oubliez de préciser une association champ / modalité (par exemple, si la réponse non a été saisie 'NoN' pour un individu) ?

Si l'INFORMAT que vous avez créé est de type caractère, dans la mesure où le champ ne prend pas plus de caractères que la plus longue des modalités finales, la réponse 'NoN' est reprise telle quelle dans votre table ; cela vous permettra de reprendre la définition de votre INFORMAT et de la compléter avec les champs que vous avez oubliés.

Si votre INFORMAT est numérique, dans le cas présent, la modalité associée dans la table sera valeur manquante, ce qui est ici beaucoup plus problématique. Lorsque vous créez une table en utilisant un INFORMAT que vous avez construit, vous devez absolument lire votre fenêtre JOURNAL. En effet, si SAS rencontre des champs non évoqués lors de la construction de l'INFORMAT, il peut vous l'indiquer de la manière suivante :

```
NOTE: Données incorrectes pour rep en ligne 521 1-11.
RÈGLE :     ----+----1----+----2----+----3----+----4----+----5----+----6-
521         oUi
rep=. _ERROR_=1 _N_=3
```

À partir de cette remarque, vous saurez comment compléter votre PROC FORMAT des associations champ / modalité pour l'instant manquantes.

b. Cas plus complexes

Dans certains cas, l'écriture complète de tous les champs possibles et des modalités associées ne sera pas possible. Vous devrez alors utiliser une procédure plus complexe. Dans un premier temps, nous reprenons le programme présenté dans la section précédente avec cette procédure plus complexe :

Programme 2.50

```
DATA infmt;
   INPUT start $11. @13 label $;
   fmtname="enq";
   type='J';
CARDS;
oui          O
Oui          O
OUI          O
non          N
NON          N
Non          N
ne sait pas NSP
NE SAIT PAS NSP
Ne Sait Pas NSP
;RUN;

PROC FORMAT CNTLIN=infmt;
RUN;

DATA test;
   INPUT nom & $10. (rep1 rep2 rep3)(& $enq.);
CARDS;
jean  oui  OUI  Oui
Pierre  non  NON  Non
paul  ne sait pas  NE SAIT PAS  Ne Sait Pas
;RUN;
```

Vous créez dans un premier temps une table dans laquelle vous résumez votre INFORMAT. Il est important que cette table contienne des variables appelées :

- START (dans laquelle vous saisirez vos champs à transformer) ;
- LABEL (modalités à donner après le passage par l'INFORMAT) ;
- FMTNAME (qui aura pour modalité le nom de votre INFORMAT) ;
- TYPE (='J' J pour INFORMAT caractère).

PROC FORMAT demande la création de l'INFORMAT au moyen des données contenues dans la table INFMT (option CNTLIN=). Nous utilisons ensuite l'INFORMAT créé pour traiter les données de la table TEST.

La variable TYPE peut prendre, en cas de création d'INFORMAT, les valeurs J (INFORMAT caractère) et I (INFORMAT numérique). Vous pourrez aussi créer des FORMAT avec ce type de programme (voir section 5.3.5) : TYPE pourra alors être égal à C si le FORMAT est caractère, et N s'il est numérique. Vous pouvez ne pas préciser TYPE si, dans le cas d'un INFORMAT numérique, vous faites précéder le nom de votre INFORMAT d'un @ ($ si l'INFORMAT est caractère)[1].

Exercice 2.24 : *On vous a transmis un fichier CHARTE.TXT qui contient des informations sur une charte graphique que vous allez devoir appliquer. Sur chaque ligne, vous disposez du nom d'un élément de cette charte graphique (ele1, ele2…) et d'une couleur notée par un code couleur X11 (voir Wikipédia, entrée « noms de couleur X11 »). Ces noms X11 (« Gainsboro », « Lemon chiffon », « Dark Slate Grey »…) ne vous sont d'aucune utilité et vous souhaitez pouvoir disposer à la place des codes hexadécimaux correspondant à ces couleurs. Créez un INFORMAT qui remplacera ces codes X11 par*

1. Un INFORMAT est caractère lorsqu'il transforme une donnée contenue dans un champ en une chaîne de caractères ; il est numérique s'il transforme la donnée contenue dans un champ en donnée numérique. Un FORMAT est numérique lorsqu'il est appliqué à une variable numérique, et caractère lorsqu'il est appliqué à une variable caractère.

les codes hexadécimaux correspondants au moment de la création de la table CHARTE à partir du fichier CHARTE.TXT. (Vous aurez très certainement besoin du fichier COULEUR.CSV dans lequel les 455 couleurs X11 sont chacune reliées à leur équivalent en code RGB et en code hexadécimal.)

Bien entendu, il est aussi possible de créer une table à partir du fichier sans créer d'INFORMAT spécifique, puis *via* une programmation, de recréer les différentes valeurs obtenues au moyen de votre INFORMAT[1]. Cependant, dans certains cas, la construction d'une table reprenant les données telles quelles, suivie d'un travail pour obtenir les modalités dans le format que vous souhaitez, sera délicate, voire impossible, particulièrement si votre fichier contient des formes de date particulières.

Imaginons le scénario suivant : dans votre fichier, vos dates ont été saisies sous la forme 1er janvier 2010, 2 janvier 2010 (des ventes sont indiquées ensuite, deux espaces séparent la date des montants vendus). Il n'existe pas d'INFORMAT qui vous permette de lire ces données. En revanche, ces dates ont un FORMAT de présentation de dates connu (voir sections 5.3 pour les FORMAT et 5.3.3 pour les FORMAT spécifiques aux dates). Retenez pour l'instant que le FORMAT est une sorte de filtre qui vous permet d'afficher, par exemple, une date dans un format lisible (et non plus comme un nombre de jours séparant votre date du 1er janvier 1960). Votre programmation devient dans ce cas :

Programme 2.51

```
DATA infmt2;
   RETAIN fmtname "datefra" type "I";
   DO label="1jan2010"d TO "31dec2012"d;
      start=PUT(label,FRADFWDX.);
      start=TRIM(LEFT(start));
      OUTPUT;
   END;
RUN;

PROC FORMAT CNTLIN=infmt2;
RUN;

DATA test;
   INPUT date & datefra. vente;
CARDS;
19 décembre 2010  1
20 décembre 2010  2
21 décembre 2011  3
22 décembre 2011  3
1er janvier 2012  5
;RUN;
```

Dans le programme 2.51, nous créons une table INFMT2 qui servira de base à l'INFORMAT. L'instruction RETAIN[2] va permettre de créer les variables FMTNAME et TYPE. La boucle DO[3] permet la création d'une série de dates entre le 1er janvier 2010 et le 1er janvier 2012. Le suffixe 'd' indique à SAS que la chaîne de texte entre les guillemets est une date (c'est pour cela que l'incrémentation va pouvoir se faire).

1. Il suffit en effet ici de saisir les champs dans une variable caractère, puis de les transformer *via* la fonction UPCASE (voir section 3.2.7) pour ensuite créer une nouvelle variable et attribuer les modalités 1, 2, 3 en fonction de la modalité observée.
2. Voir section 3.6.1. Cette instruction n'est pas indispensable ici, on pouvait tout aussi bien créer ces mêmes variables au moyen de deux instructions :
   `fmtname="datefra" ; type="I";`
3. Voir section 3.5.2.

La fonction PUT[1] crée une variable caractère START égale à la date écrite dans le FORMAT FRADFWDXw. (qui est bien la forme que prennent les dates dans notre fichier). La fonction TRIM permet d'effacer les espaces qui pourraient apparaître dans l'enregistrement de la date : vous forcez le positionnement de la date à gauche et si des espaces apparaissent, ils sont effacés.

L'instruction OUTPUT permet à chaque tour dans la boucle d'écrire les variables LABEL, START, FMTNAME et TYPE. Ensuite, la procédure PROC FORMAT, comme dans l'exemple précédent, crée l'INFORMAT spécifique que vous pouvez ensuite appliquer pour créer votre table. Là encore, une programmation à l'intérieur d'une étape DATA est possible, mais nettement plus délicate que la programmation proposée ici.

Nous avons dans le cas présent profité du fait que, même si l'INFORMAT n'existait pas, la date était affichée dans un FORMAT connu de SAS. Mais vous rencontrerez bien souvent des formats de dates qui ne ressembleront à rien de connu. Imaginons par exemple que la date soit notée sous la forme 2006/AOU/01 dans votre fichier.

Quels sont les problèmes liés à cette date ?

- Effectivement, aucun INFORMAT ne peut lire cette date.
- Cette date n'est même pas présentée dans un FORMAT de SAS.
- Elle est en français (AOU pour août ; si elle avait été en anglais, on aurait dû lire AUG).

Vous pouvez vous en sortir avec une programmation légèrement modifiée :

Programme 2.52

```
OPTIONS dflang=french;

PROC FORMAT;
   PICTURE tempo low-high='%Y/%b/%d' (DATATYPE=date);
RUN;

DATA infmt3;
   RETAIN fmtname "datefraa" type "I";
   DO label="01jan2004"d TO "01jan2007"d;
      start=PUT(label,tempo11.);
      start=TRIM(LEFT(start));
      OUTPUT;
   END;
RUN;

PROC FORMAT CNTLIN=infmt3;
RUN;

DATA test;
   INPUT date datefraa.;
CARDS;
2006/aou/15
2005/sep/21
2004/jan/5
;RUN;
```

1. Voir section 5.3.6.

Nous choisissons le français comme langue par défaut *via* OPTIONS puisque les abréviations des mois sont en français. Le premier PROC FORMAT va créer un FORMAT (tempo) *via* l'instruction PICTURE[1]. Vous indiquez ainsi à SAS que toutes les valeurs (low-high) sont organisées de la façon suivante : %Y : année en 4 chiffres ; %b : mois en trois lettres dans la langue par défaut ; %d : jour du mois en chiffres (voir plus bas pour d'autres codes possibles). Année, mois et jour sont séparés par des '/' (barres obliques) ; l'option DATATYPE= indique que nous créons bien un INFORMAT de date.

Ensuite, vous créez une table INFMT3 qui vous servira à construire votre INFORMAT en suivant la même procédure que précédemment. La seule difficulté est de donner à SAS le nombre de caractères qu'il doit utiliser pour écrire la date dans le FORMAT temporaire – ici, 11 au maximum, 4 pour l'année + 1 (/) + 3 pour le mois + 1 (/) + 2 pour le jour.

L'instruction PICTURE sur des dates ayant des formes très particulières vous permettra de vous sortir de très nombreuses situations délicates. Différents codes sont possibles :

- %A : nom du jour en toutes lettres.
- %b : nom du mois en abrégé.
- %B : nom du mois en toutes lettres.
- %d : numéro du jour dans le mois.
- %j : numéro du jour dans l'année.
- %m : mois en décimal.
- %w : jour de la semaine en décimal.
- %y : année en deux chiffres (sans les siècles).
- %Y : année en quatre chiffres (avec les siècles).

Ces instructions fonctionneront aussi sur des dates dans des langues autres que l'anglais (mais vous devrez spécifier la langue à utiliser *via* OPTIONS). Si la manœuvre peut sembler compliquée, elle est cependant nécessaire : PICTURE ne peut en effet créer que des FORMAT.

Exercice 2.25 : *Vous avez dans votre fichier la date suivante : « Mercredi 12 décembre 2007 ». Adaptez la programmation pour créer l'INFORMAT qui saisira cette date. Imaginez maintenant que cette date soit de la forme : « mercredi 12 décembre 2007 ». Adaptez à nouveau votre programmation pour saisir correctement cette date. Et que devient votre programmation si votre date est au format : « Mercredi, le 12 décembre 2007 » ?*

Exercice 2.26 : *Dans le fichier ZODIAQUE.TXT, vous disposez d'informations sur des individus nés en 1990 (prénom et date de naissance sous la forme DDMMYY10). Créez un INFORMAT qui permettra d'interpréter la date de naissance pour pouvoir disposer dans votre table du signe zodiacal de la personne. (Une consultation, par exemple de Wikipédia, entrée « signe du zodiaque » vous sera très utile.) Vous pouvez suivre les deux méthodes exposées dans cette section pour construire cet INFORMAT.*

Exercice 2.27 : *Adaptez votre programme de façon à ce que les signes zodiacaux soient présentés dans la table que vous créerez à partir du fichier zodiaque2.txt. Ce fichier contient les données d'individus nés entre 1985 et 1995. Vous n'avez besoin d'ajouter que quelques instructions au programme réalisé à l'occasion de l'exercice 2.26.*

Les exercices 2.25, 2.26 et 2.27 sont de difficulté croissante. Pour traiter les problèmes posés, il est nécessaire de bien maîtriser les outils qui sont étudiés au chapitre 3. Vous avez beaucoup à apprendre de ces exercices et les solutions proposées sur www.sas-sr.com font l'objet de commentaires particuliers.

1. Voir section 5.3.5.b pour plus de détails sur la création de FORMAT au moyen de l'instruction PICTURE.

2.7 SAS, Excel, l'importation et l'exportation de tables

Il vous arrivera souvent d'utiliser SAS pour analyser les données contenues dans des fichiers issus d'autres logiciels ou pour envoyer les données d'une table dans un fichier lisible par un autre logiciel. Nous traiterons dans cette section des outils que SAS met à disposition pour importer des données, puis des outils d'exportation de tables.

2.7.1 Le module d'importation – PROC IMPORT

Dans un premier temps, si vous souhaitez importer des données dans SAS, vous pouvez recourir au module d'importation. Celui-ci est accessible à partir du menu fichier (voir figure 2.7).

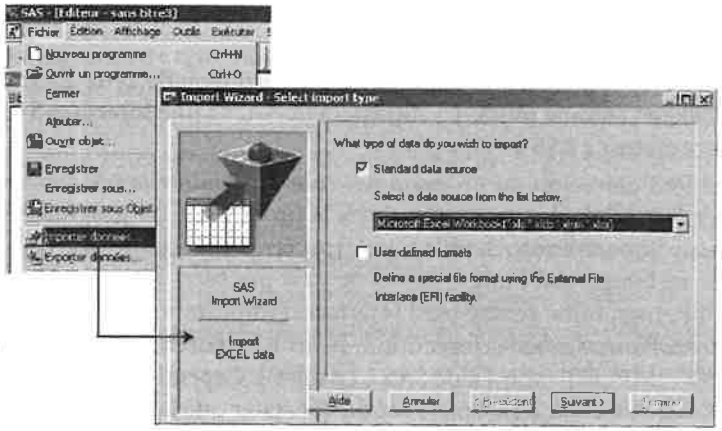

Figure 2.7 • Le module d'importation.

Au moyen de ce module d'importation, vous pouvez importer des fichiers créés par Excel (2007 et versions précédentes), Access (97 et 200x), CSV, TXT, dBASE, JMP, SPSS, STATA, PARADOX et LOTUS 1-2-3.

Lorsque vous aurez sélectionné le type de fichier, différents écrans s'afficheront au fur et à mesure de vos actions. Si vous importez un classeur Excel, vous aurez dans l'ordre :

- Premier écran : vous naviguez sur votre disque dur et sélectionnez le classeur Excel à importer.
- Deuxième écran : vous sélectionnez la feuille à importer (vous ne pouvez importer qu'une feuille Excel à la fois).