a Integration Server» et «SAS Enterprise Data Integration Server».
w.sas.com/france/software/technologies/dw.html
S Forecast Server est une application d'aide à la production de prévisions.
w.sas.com/technologies/analytics/forecasting/forecastserver/
S/IML Studio est une interface experte avancée pour le statisticien maîtrisant ploration graphique comme la programmation SAS. Elle permet de recourir à la istique exploratoire tout en se concentrant sur le développement de nouveaux rithmes (approche «statistiques-mathématiques») et nécessite la maîtrise de la grammation SAS/STAT et SAS/IML.
://support.sas.com/rnd/app/studio/studio.html

n, citons :

NLS (*National Language Support*), qui n'est pas un module en tant que tel mais application qui permet à SAS de fonctionner au mieux sur des systèmes non ricains.

Créer une table SAS

Au sommaire de ce chapitre :

- © 1.1 Premiers pas avec SAS
- © 1.2 Deux étapes distinctes : DATA et PROC
- © 1.3 Les tables SAS
- © 1.4 Cas simple : informations séparées par un espace
- © 1.5 Compréhension appronfondie du cas simple
- © 1.6 Introduction de valeurs manquantes
- © 1.7 Variables numériques et variables caractères

Ce premier chapitre a pour objectif de vous aider dans vos débuts avec SAS. Il s'agit plus particulièrement de vous présenter l'environnement de développement SAS et de vous apprendre à créer vos premières tables.

© 1.1 Premiers pas avec SAS

Au démarrage de SAS, l'écran de la figure 1.1 apparaît.

Figure 1.1 • Capture d'écran SAS Windows.

Les étapes DATA ont généralement pour objectif :

- soit de créer, à partir d'un fichier de données, une table SAS exploitable par la suite (voir la première étape DATA de la figure 1.3) ;
- soit de modifier une table SAS existante ou, à partir d'une table existante, de créer une nouvelle table SAS (voir la seconde étape DATA de la figure 1.3).

Les étapes PROC permettent d'exploiter les données contenues dans des tables SAS par l'application de protocoles préécrits et propres à chaque procédure. Ces étapes peuvent conduire :

- à la production d'une nouvelle table SAS (table de résultats – voir la première étape PROC de la figure 1.3) ;
- à la production d'un rapport dans la fenêtre SORTIE, par exemple (voir la seconde étape PROC de la figure).

En dehors de ces deux étapes, votre programme peut aussi contenir des **instructions globales**. Celles-ci agissent sur votre environnement de travail et sont effectives tout au long de votre session ou jusqu'à ce que vous les annuliez. Nous avons utilisé à la figure 1.3 l'instruction globale de définition d'une bibliothèque (LIBNAME – voir section 1.4.4). Les instructions telles que OPTIONS (voir section 5.7), TITLE et FOOTNOTE (voir section 6.1.2) sont d'autres instructions globales possibles.

Utiliser SAS consiste avant tout à écrire des programmes : il convient ainsi de vous fournir quelques éléments de langage.

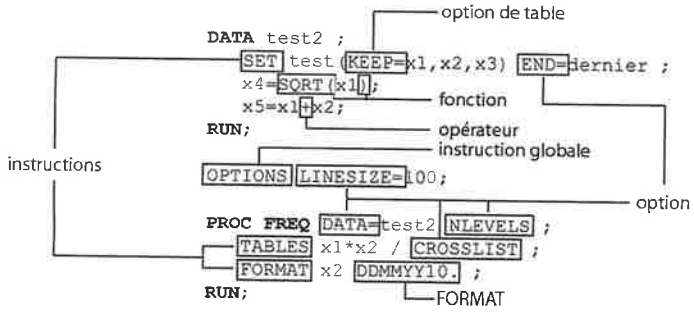


Figure 1.4 • Éléments de langage.

Au sein de chaque étape (DATA ou PROC), vous employez des **instructions** qui mobiliseront des variables ou des tables. Ces instructions se trouvent généralement en début de ligne et sont souvent accompagnées d'**options** précisées ensuite, parfois précédées du signe /. Attention cependant : tout ce qui n'est pas une instruction ou un nom de variable n'est pas forcément une option. Ainsi, DDMMYY10. est un FORMAT. On peut construire les variables en faisant appel à des **fonctions** (qui ont un ou plusieurs **arguments**) ou à des **opérateurs**. Ceux-ci sont toujours suivis d'un nom de variable, d'une constante numérique, d'une modalité caractère, d'une fonction, etc. Les **options de table** sont précisées entre parenthèses et suivent le nom d'une table. Vous pouvez y recourir dès qu'un nom de table apparaît.

1.3 Les tables SAS

Dans le vocabulaire SAS, le terme « table » indique le fichier dans lequel sont stockées les données que vous allez analyser. Une table SAS présente trois caractéristiques :

- Une table est nécessairement présente dans une bibliothèque SAS.
- Elle contient des données organisées sous la forme d'un tableau dans lequel les colonnes présentent des variables (âge, sexe, diplôme…), et les lignes présentent des observations (individus, périodes…).
- Elle contient aussi des informations descriptives de la table (métadonnées) : type des variables, longueur de celles-ci, nombre d'observations…

En ce qui concerne la dimension « données » d'une table SAS, si vous ne regardez qu'une colonne, vous verrez l'ensemble des modalités que peut prendre une variable au sein de votre population. Sur une ligne, vous verrez les modalités prises par toutes les variables pour une observation donnée.

Afin de mieux comprendre la structure d'une table de données, nous avons ouvert au moyen de l'onglet EXPLORATEUR la table CLASS, livrée avec SAS 9.1 et 9.2. Pour ouvrir cette table, allez dans l'onglet EXPLORATEUR, double-cliquez sur BIBLIOTHEQUES, puis sur SASHELP et enfin sur CLASS (voir figure 1.5).

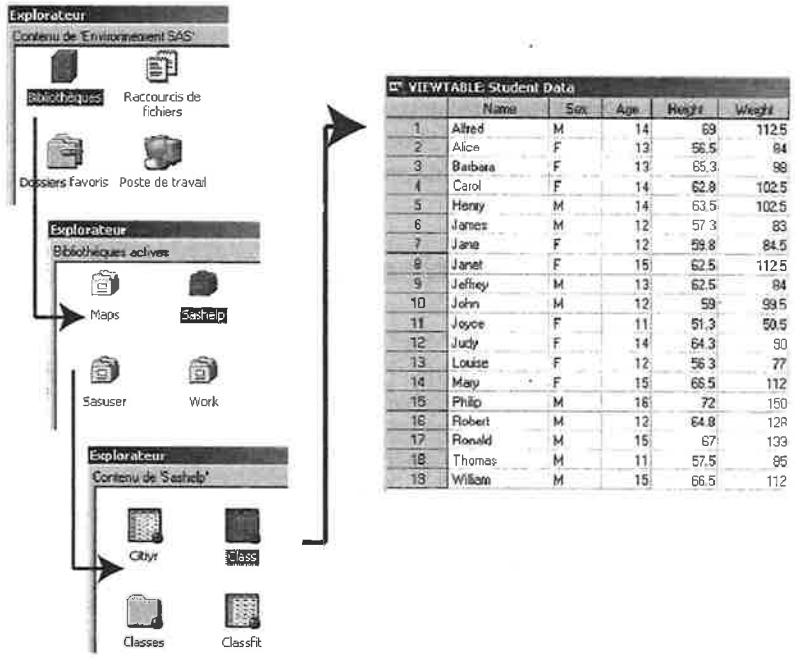


Figure 1.5 • Ouvrir une table dans la fenêtre VIEWTABLE.

Les variables contenues dans cette table sont NAME, SEX, AGE, HEIGHT et WEIGHT. Les observations sont relatives à des individus. Cette table en contient 19.

Il ne faut pas confondre cet éditeur de table avec un tableur. Il est possible d'apporter certaines modifications à la table *via* la fenêtre VIEWTABLE (modification de certaines valeurs, tris, mises en forme…), mais pour toute modification majeure, vous devrez passer par une étape DATA et donc par l'écriture d'un programme dans la fenêtre EDITEUR.

Pour accéder à la zone descriptive de cette table, vous pouvez au choix :

- demander les propriétés de la table en cliquant du bouton droit, dans l'onglet EXPLORATEUR, sur l'icône de la table CLASS ;
- demander un PROC CONTENTS sur cette table (voir section 5.1).

Concentrons-nous à présent sur la phase de création des tables SAS. C'est une étape extrêmement importante puisqu'elle conditionne votre résultat final : votre table doit absolument être le reflet exact des données dont vous disposez actuellement en dehors de SAS. Si tel n'était pas le cas, les analyses que vous pourriez mener par la suite n'auraient aucune valeur.

SAS offre des outils qui vous permettent de vous assurer que votre table est bien une fidèle reproduction de vos données originales. Ces divers outils seront présentés dans la section 2.8.

Nous allons voir dans la section suivante comment, dans le cas le plus simple, créer une table SAS à partir de données brutes. Ces données que vous souhaitez voir transformées en une table SAS peuvent prendre deux formes : soit elles se trouvent déjà dans un fichier informatique, soit elles sont sur une feuille de papier et vous devez les saisir dans la fenêtre EDITEUR.

C'est ce dernier cas que nous présentons en premier, pour sa simplicité – mais vous le rencontrerez très rarement. Vous pourrez ainsi vous familiariser avec les tables SAS et comprendre comment SAS interprète les données qui lui sont soumises.

© 1.4 Cas simple : informations séparées par un espace

Dans cette section, nous allons créer nos premières tables SAS. Au-delà de ce travail, il s'agit aussi d'approfondir notre connaissance du fonctionnement de SAS et son environnement. L'exécution de notre premier programme nous permettra de nous familiariser avec l'EDITEUR SAS et les bibliothèques.

1.4.1 Créer une première table SAS

Programme 1.1

```
DATA test;
   INPUT x1 x2 x3 x4 x5 $;
CARDS;
78 5 5 1161 ABÉLIEN
35 6 3 1336 ALMÉRIC
82 1 5 1499 ANIEL
;RUN;
```

Un point de terminologie avant de commencer l'analyse de ce programme : notre objectif est de construire une **table** qui indiquera, pour chaque **observation**, les **modalités** prises par diverses **variables**. Nous partons pour cela de **données** présentées soit à l'écran, soit dans un **fichier** externe à SAS. Un **enregistrement** correspond à une ligne de données dans votre fichier et il est composé de **champs**. On distingue les différents champs au moyen des **séparateurs de champs**. **L'espace** est le séparateur de champs par défaut.

Nous sommes ici dans le cas le plus simple : il y a cinq variables à saisir (X1 X2 X3 X4 X5) et vous observez qu'il existe, par enregistrement, cinq champs séparés par des espaces[1] (78 5 5 1161 ABÉLIEN pour le premier enregistrement). Chaque enregistrement présente les modalités prises par les variables pour une observation (et une seule).

```
DATA test;
```

Cette instruction demande à SAS de créer une table que l'on va appeler TEST. Comme **TOUTES** les instructions de SAS, celle-ci se termine par un « ; ». Si vous oubliez de conclure une instruction par un point-virgule, le programme ne s'exécutera pas. Au début, 80 % des erreurs de programmation que vous rencontrerez seront liées à des « ; » oubliés.

Le nom d'une table ne doit pas excéder 32 caractères. Vous pouvez utiliser toutes les lettres **non accentuées** et le signe « _ ». Il peut contenir des chiffres mais ne doit pas débuter par un chiffre. SAS ne fait pas la distinction entre les majuscules et les minuscules. La table TEST est donc parfaitement équivalente à la table 'test'. Si, au moyen de l'onglet EXPLORATEUR, vous recherchez votre table TEST, celle-ci est placée dans la bibliothèque WORK et a pour nom 'Test'. Pour écrire le programme, vous pouvez employer aussi bien les majuscules que les minuscules. La mise en forme du programme n'a aucune importance : vous pouvez ajouter autant d'espaces ou de sauts de lignes que vous le souhaitez entre, par exemple, les termes DATA et TEST.

```
INPUT x1 x2 x3 x4 x5 $;
```

Dans cette table TEST, il y a cinq variables : X1, X2, X3 et X4 sont des variables numériques, X5 est une variable caractère (du texte) – on indique à SAS qu'une variable est caractère en ajoutant le signe « $ » après son nom. INPUT est une instruction essentielle puisqu'elle permet à SAS de comprendre la structure des données qu'il va devoir transformer en une table SAS. Nous disposons, comme nous le verrons par la suite, d'un ensemble d'options qui indiquent la structure et la nature de ces données, et comment elles doivent être interprétées pour pouvoir être utilisées dans une table SAS.

Vos variables peuvent prendre à peu près n'importe quel nom. Celui-ci ne doit pas dépasser 32 caractères. Le nom de votre variable peut contenir des lettres **non accentuées**, des chiffres et le signe « _ ». En revanche, les signes %, $, !, *, &, #, @ sont à proscrire. Le nom d'une variable ne peut pas commencer par un chiffre.

SAS ne fait aucune différence entre les majuscules et les minuscules dans les noms de variables. Ainsi, il ne distingue pas les variables TOTO et toto, ce qui implique que vous ne pouvez pas, au sein d'une même table, créer à la fois une variable X1 et une variable x1.

1. Si vos champs sont séparés par plusieurs espaces, cela ne change rien. Si vous ajoutez des espaces dans le programme 1.1, par exemple, la table sera parfaitement créée.

au moment où vous créez une table, vous déclarez la création d'une variable « ToTo », SAS écrira systématiquement « ToTo » dans ses sorties lorsqu'il y aura invocation de cette variable ; mais vous pourrez appeler sans aucune difficulté dans vos programmes cette variable au moyen de toto, TOTO, tOtO, TotO, etc.

Depuis la version 9 de SAS, il n'existe aucune limite en dehors de celles que vous impose l'ordinateur que vous utilisez quant au nombre de variables présentes dans une table SAS. Le nombre d'observations que vous pouvez inclure dans une table SAS est lui aussi illimité. Une fois de plus, seules les capacités de votre ordinateur limiteront le nombre d'observations de votre table.

Le séparateur décimal interne à SAS est le point (3.14). Si vous ne précisez pas l'INFORMAT (voir section 2.6) qui permet de faire comprendre à SAS des données avec le séparateur décimal virgule, « 3,14 » ne sera pas compris comme un nombre.

```
CARDS;
```

Au moyen de cette instruction, vous indiquez ici à SAS que les données arrivent. Vous pouvez aussi utiliser l'instruction DATALINES.

```
78 5 5 1161 ABÉLIEN
35 6 3 1336 ALMÉRIC
82 1 5 1499 ANIEL
```

Voici donc les données : elles sont présentées en lignes. Pour le premier individu X1=78, X2=5, X3=5, X4=1161 et X5 (variable caractère)=ABÉLIEN. C'est lorsque SAS rencontre le séparateur de champs (ici l'espace), qu'il sait que la modalité de la variable est terminée et qu'il peut entamer la lecture de la modalité de la variable suivante.

```
;RUN;
```

Nous avons déjà indiqué qu'une instruction SAS prend généralement fin avec un « ; ». Lorsque l'on constitue une table SAS, il est important de situer le point-virgule qui marque la fin de l'arrivée des données sur la ligne qui suit le dernier enregistrement. Vous perdez ce dernier enregistrement si, au lieu de

```
82 1 5 1499 ANIEL
;RUN;
```

vous entrez :

```
82 1 5 1499 ANIEL;
RUN;
```

Vous remarquerez que, dans ce cas, le fond jaune caractéristique des plages de saisie des données dans l'éditeur amélioré a disparu et que les données numériques apparaissent en sarcelle (bleu-vert moyen) et en gras. En général, SAS n'accorde aucune importance aux retours chariot ; faites cependant attention au retour chariot dans les parties de programmes où sont présentées les données. Dans certains cas, vous pourriez avoir une table différente de vos données originales.

L'instruction RUN qui clôt le programme n'est pas obligatoire si vous utilisez une instruction CARDS. Votre programme sera parfaitement exécuté sans celui-ci, mais vous devrez le soumettre jusqu'au point-virgule placé sur la ligne après le dernier enregistrement. De façon générale, l'instruction RUN indique à SAS que tous les éléments propres à l'étape sont maintenant définis et qu'en cas de demande d'exécution, il peut exécuter les tâches demandées par l'étape.

1.4.2 L'éditeur SAS : une aide à la programmation

SAS est conscient que son langage de programmation peut être compliqué et vous aide en affichant certains mots dans des couleurs particulières. Il est important d'être attentif aux couleurs que prennent les différents éléments de votre programme puisqu'elles peuvent, avant même que vous ne soumettiez votre programme, vous indiquer si celui-ci à des chances d'être exécuté correctement ou pas.

La lecture seule de ce livre en noir et blanc ne vous éclairera pas beaucoup quant aux différentes couleurs que vous rencontrerez dans votre éditeur. Merci de reproduire dans votre éditeur SAS le programme 1.2[1].

Programme 1.2

```
* programme de création de la table test;
/* le programme */
DATA test;
   INPUT x1 8.4 x2 DDMMYY.;
   INFILE 'c:\mes documents\test.txt';
   VAR x1;
   IMPUT x2;
   IF x1=1 THEN z1="texte";
   FORMAT x1 8.4;
CARDS;
1 2
3 4
;RUN;

PROC PRINT DATA=test;
   VAR x1;
RUN;

%MACRO print;
   %LET texte=test;
   PROC PRINT;RUN;
%MEND;

%print
```

En vert apparaissent les commentaires que vous avez saisis dans vos programmes. Ils sont introduits :

- au moyen d'un astérisque (*) – le commentaire prend alors fin au premier « ; » rencontré ;
- au moyen de « /* » – le commentaire prend alors fin lorsque vous entrez la chaîne « */ ».

Les mots clés ouvrant et fermant les étapes (DATA/RUN, PROC/RUN et %MACRO/%MEND) apparaissent en bleu et en gras. À l'intérieur des étapes DATA et PROC, les instructions propres à l'étape sont reconnues par l'éditeur et s'affichent en bleu clair (INPUT, INFILE, IF, THEN, FORMAT, CARDS pour l'étape DATA ; VAR et DATA pour l'étape PROC PRINT). Les instructions VAR et IMPUT de l'étape

1. Vous pouvez aussi télécharger ce programme sur le site compagnon de cet ouvrage : **www.sas-sr.com**. Ce programme est proposé à titre d'illustration. Il n'a en fait auccun sens.

DATA ne sont pas reconnues par l'éditeur et apparaissent en rouge dans le programme (VAR parce que cette instruction n'est pas possible dans l'étape DATA et IMPUT parce que cette instruction n'existe pas – les erreurs de frappe dans les noms des instructions peuvent donc être repérées).

Les chaînes de caractères entre quotes, simples ou doubles, sont reprises en violet. Elles indiquent soit un chemin vers un fichier externe, soit la modalité d'une variable caractère. La zone de saisie des données est présentée avec un fond jaune. Les noms des INFORMAT (DDMMYY.), des FORMAT et des constantes numériques (1 dans la ligne débutant par IF) apparaissent en sarcelle (bleu-vert moyen). Les constantes numériques apparaissent de plus toujours en gras ; c'est également le cas de certains FORMAT ou INFORMAT (de type W.D : 8.4)[1].

Les autres éléments des programmes des étapes DATA et PROC apparaissent en noir : il s'agit des noms des tables créées ou utilisées (TEST) et des noms des variables (X1, X2, Z1).

La programmation macro présente des singularités : le texte à l'intérieur des macro-programmes apparaît généralement en noir non gras, sauf les éléments propres au langage macro. Ces éléments débutent habituellement par le signe % (%LET dans le programme 1.2) et apparaissent en bleu.

Il est donc important d'observer les couleurs que prend le texte. Si un élément de votre programme apparaît en rouge, il est parfaitement inutile de soumettre celui-ci[2]. Si le mot PROC ne devait pas apparaître en bleu et en gras, relisez votre programme et trouvez-en la raison avant toute demande d'exécution (à titre d'illustration, retirez le point-virgule placé juste avant PROC PRINT).

```
⊟DATA test;
    INPUT x1 x2 x3 x4 x5 $;
CARDS;
78 5 5 1161 ABÉLIEN
35 6 3 1336 ALMÉRIC
82 1 5 1499 ANIEL
;RUN;
```

Figure 1.6 • Capture d'écran du programme 1.1.

La figure 1.6 présente une capture d'écran du programme 1.1. Dans l'éditeur, un signe ⊟ apparaît systématiquement à gauche de l'instruction DATA ou PROC de début d'étape. En cliquant sur ce symbole, vous avez la possibilité de réduire ou développer chaque étape DATA ou PROC.

1.4.3 Demander l'exécution d'un programme

Une fois le programme saisi, il convient de demander à SAS de l'exécuter. Cela peut être fait de plusieurs façons mais dans tous les cas, vous devez au préalable sélectionner votre programme de la même manière que vous sélectionnez un texte dans

[1] INFORMAT sont traités dans la section 2.6, les FORMAT sont présentés dans la section 5.3.

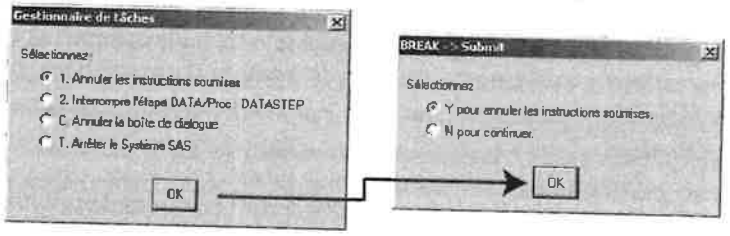
votre traitement de texte. Pour demander l'exécution de votre programme, vous avez plusieurs possibilités :

- Vous appuyez sur la touche F3 de votre clavier.
- Vous cliquez sur l'icône située dans la barre d'outils.
- Vous sélectionnez la commande **Soumettre** du menu **Exécuter**.
- Vous cliquez du bouton droit et choisissez dans la zone de l'éditeur **Soumettre sélection** ou **Soumettre tout**.
- Pour les programmes qui utilisent beaucoup de ressources, vous pouvez également demander une exécution en mode BATCH (voir figure 1.7). Vous devez saisir votre programme, puis l'enregistrer sur votre disque dur. Terminez votre session SAS (quittez SAS) et, *via* l'explorateur Windows, retrouvez votre programme sur votre disque dur. Cliquez du bouton droit et sélectionnez **Batch Submit with SAS 9.2**.

Figure 1.7 • L'exécution d'un programme en mode Batch.

Si vous ne sélectionnez pas votre programme et si vous demandez une exécution, toutes les lignes de programme saisies dans la fenêtre EDITEUR seront exécutées (même les parties de programme que vous ne voulez pas exécuter). Si votre programme disparaît (c'est possible avec certaines versions de SAS), appuyez sur la touche F4 ou demandez dans le menu **Exécuter/Rappeler la dernière exécution** pour le faire réapparaître.

Si, pour une raison ou une autre, vous souhaitez interrompre le programme, vous pouvez soit appuyer simultanément sur Ctrl et Pause, soit sur le bouton dans la barre d'outils. Dans tous les cas, la fenêtre n° 1 présentée à la figure 1.8 apparaîtra.



Fenêtre n°1 Fenêtre n°2

Figure 1.8 • Interrompre un programme SAS.

Le fait de cliquer sur 'OK' dans la fenêtre n° 1 ouvre la fenêtre n° 2, dans laquelle vous confirmez l'annulation de l'exécution de votre programme.

Vous pouvez à présent soumettre à SAS le programme 1.1 présenté précédemment.

Rien ne semble se produire, mais c'est une fausse impression. Passez dans la fenêtre JOURNAL. Si SAS a correctement exécuté votre programme, vous devez le voir s'afficher en noir accompagné de commentaires. Tant que ceux-ci sont en bleu, tout va bien. Si vous voyez du vert, c'est qu'il y a eu un «petit» problème : même si SAS a exécuté votre programme, il se pose des questions quant à la pertinence de celui-ci. Si vous voyez du rouge, c'est qu'une erreur de programmation majeure a eu lieu. Le programme n'a pas été exécuté parce qu'il n'a pas pu être compilé (voir section 3.1).

Lorsque vous examinez votre fenêtre JOURNAL, commencez à lire **à partir du début du programme** : une erreur qui apparaît en fin de programme peut être due à une erreur intervenue plus tôt. Si vous vous focalisez uniquement sur la dernière erreur, il est bien possible que vous ne trouviez jamais pourquoi le programme n'a pas fonctionné. Ces erreurs qui empêchent la compilation sont généralement de trois types :

- Vous avez mal orthographié certains mots clés.
- Des «;» sont manquants ou non valides.
- Vous faites appel à des options ou à des instructions non valides.

Vous devez ensuite systématiquement vérifier que votre table est bien conforme aux données originales. Si votre table ne comprend pas trop d'observations, vous pouvez utiliser la procédure PROC PRINT :

```
PROC PRINT DATA=test;
RUN;
```

Chaque procédure de SAS effectue une tâche bien précise. La procédure PRINT demande par exemple l'impression de la table. Maintenant, quelle que soit la procédure employée, vous aurez toujours le même type de syntaxe :

```
PROC XXXXX DATA=xxxxx options;
  instructions;
RUN;
```

Vous pouvez ne pas indiquer, au moyen de DATA=, la table sur laquelle vous voulez exécuter cette procédure. Elle sera alors exécutée sur la dernière table créée.

Sur votre fenêtre SORTIE apparaît la table telle que SAS l'a comprise. Pour limiter le nombre d'observations (imaginez ce qui se passerait si votre table comportait quelques millions d'observations), vous pouvez spécifier le nombre d'observations à envoyer dans la fenêtre SORTIE.

```
PROC PRINT DATA=test(OBS=10);
RUN;
```

Dans le cas présent, les dix premières observations seront envoyées dans votre fenêtre SORTIE. Si vous souhaitez voir les dix dernières observations, regardez dans un premier temps le nombre d'observations de votre table. Celui-ci est indiqué dans la fenêtre JOURNAL lors de la création de votre table. Imaginons que celui-ci soit égal à 150000. Ensuite[1] :

```
PROC PRINT DATA=test(FIRSTOBS=149991);
RUN;
```


Pour imprimer les observations 9990 à 10000 (soit 11 observations) :

```
PROC PRINT DATA=test(FIRSTOBS=9990 OBS=10000);
RUN;
```

La valeur donnée à OBS est nécessairement supérieure à celle donnée à FIRSTOBS. Les options OBS= et FIRSTOBS= ne sont pas des options de PROC PRINT mais des options de table (*DATA set options*). Vous pouvez y recourir dès qu'un nom de table est cité dans une procédure ou dans une étape DATA[1].

1.4.4 Les bibliothèques

Si tout se passe bien lors de la phase de création de la table, SAS vous donne ce type de message dans la fenêtre JOURNAL :

```
NOTE: The data set WORK.TEST has 3 observations and 5 variables.
```

Cela signifie qu'il a créé une table TEST dans la bibliothèque WORK. Par défaut, vous disposez de quatre bibliothèques : WORK, SASHELP, SASUSER et MAPS[2] (voir figure 1.9). Pour accéder à leur contenu, cliquez sur l'onglet EXPLORATEUR, puis double-cliquez sur BIBLIOTHEQUES.

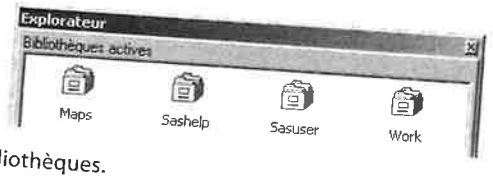


Figure 1.9 • Les bibliothèques.

La bibliothèque MAPS contient les fonds de cartes que vous pouvez utiliser dans la procédure graphique GMAP. La bibliothèque SASHELP contient les tables qui vous permettent de refaire les exemples proposés dans l'aide. Elle contient également un certain nombre de fichiers nécessaires au bon fonctionnement de SAS. Vous ne pouvez rien écrire dans ces bibliothèques.

La bibliothèque SASUSER contient les informations qui sont propres à l'utilisateur. WORK est la bibliothèque temporaire, vidée de son contenu lorsque vous quittez SAS. C'est dans cette bibliothèque que seront placées, par défaut, les tables que vous allez créer au cours d'une session. Si vous souhaitez conserver ces tables pour une utilisation ultérieure, vous devez les placer dans une bibliothèque permanente.

L'instruction DATA TEST est parfaitement équivalente à l'instruction DATA WORK.TEST. Vous trouverez dans l'aide SAS ou sur Internet de très nombreux programmes dans lesquels les auteurs précisent systématiquement la bibliothèque WORK pour indiquer que la table est à placer dans la bibliothèque temporaire. Dans cet ouvrage, si aucune bibliothèque n'est indiquée, c'est que la table créée ou modifiée est située dans la bibliothèque temporaire.

Pour créer une bibliothèque permanente, vous pouvez utiliser l'instruction suivante :

```
LIBNAME lib 'c:\mes documents';
```

1. Pour plus de détails

LIBNAME : demande à SAS de créer une bibliothèque permanente. Le nom de cette bibliothèque est LIB. 'c:\mes documents\' est l'emplacement physique, sur votre disque dur, de cette bibliothèque. L'instruction LIBNAME fait partie des instructions qui définissent votre environnement. C'est une **instruction globale** ; elle se place généralement en dehors des instructions DATA et PROC. Le chemin physique doit être encadré par des quotes, simples ou doubles. Cependant, si dans le nom de l'un de vos dossiers cités apparaît une quote, vous devrez impérativement utiliser des quotes doubles :

```
LIBNAME toto "C:\intro_SAS\fichiers pour l'analyse";
```

La seule difficulté consiste ici à donner à SAS un chemin DOS correct. Si votre chemin n'est pas correct, la bibliothèque ne peut pas être créée. Il est possible d'attribuer à une bibliothèque plusieurs emplacements. Ainsi :

Programme 1.3

```
LIBNAME lamar ('d:\mes documents\sas exemple' 'c:\mes documents');
```

Vous pouvez aussi créer une bibliothèque en utilisant l'icône « Nouvelle bibliothèque » (voir figure 1.10) de la barre d'outils.

Figure 1.10 • Le bouton Nouvelle bibliothèque.

Autre possibilité : cliquez du bouton droit dans l'onglet EXPLORATEUR actif, puis sélectionnez **Nouveau**. Dans les deux cas, la fenêtre présentée à la figure 1.11 apparaîtra.

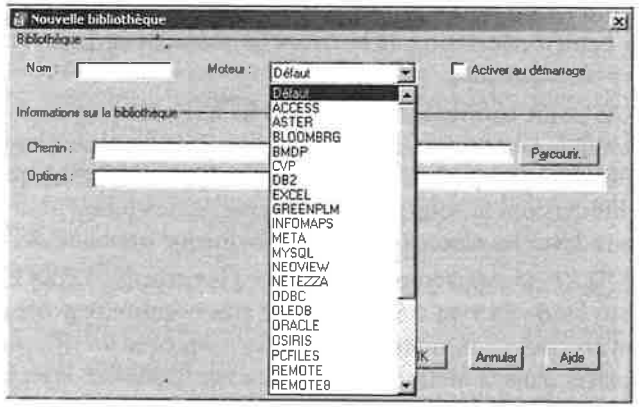


Figure 1.11 • Création d'une nouvelle bibliothèque.

Vous devez alors préciser le nom, le moteur et le chemin physique de votre bibliothèque. En spécifiant un moteur, vous avez la possibilité d'indiquer à SAS quel type de fichiers il va trouver à l'emplacement physique que vous lui indiquez. Le moteur par défaut est V9 (pour SAS, version 9)[1]. En cochant **Activer au démarrage**, la bibliothèque que vous créez sera systématiquement définie à chaque début de session. Si vous ne cochez pas **Activer au démarrage**, la bibliothèque sera détruite à la fin de votre session SAS. **Attention**, cela ne signifie pas que toutes les tables créées et placées dans cette bibliothèque seront détruites, mais que l'association entre le nom de votre bibliothèque et son emplacement physique sur votre disque dur sera perdue. Vous aurez toujours la possibilité, au cours d'une session ultérieure, de réallouer à cet emplacement physique une bibliothèque, de même nom ou pas, et d'analyser à nouveau vos tables sauvegardées.

Le nom d'une bibliothèque ne doit pas excéder 8 caractères. Vous pouvez utiliser toutes les lettres non accentuées et le signe « _ ». Il peut contenir des chiffres mais ne doit pas débuter par un chiffre.

Si vous avez à spécifier un moteur particulier, vous pouvez aussi le faire au moyen de l'instruction LIBNAME.

```
LIBNAME old V6 'C:\intro_SAS\tables V6 de SAS';
```

Dans tous les cas, il conviendra de regarder la fenêtre JOURNAL afin de voir si votre bibliothèque a été correctement créée. Parmi les messages que vous trouverez dans la fenêtre JOURNAL, vous observerez le message suivant :

```
Moteur :        V9
```

Cela signifie qu'à cet emplacement, seront à lire et seront écrites des tables enregistrées dans le format propre aux versions 7 (très peu distribuée), 8 et 9 de SAS[1]. Parmi les moteurs possibles, vous remarquez la présence d'un moteur Excel. Vous devinez donc qu'il est possible de lire et d'écrire directement au moyen de SAS des fichiers Excel. Nous traiterons de la construction de tables SAS à partir de classeurs Excel dans les sections 2.7.1 et 2.7.2.

Pour placer la table TEST dans votre bibliothèque, il vous suffit d'exécuter le programme 1.4.

Programme 1.4

```
DATA toto.test;
   SET test;
RUN;
```

La première instruction demande la création d'une table TEST dans la bibliothèque TOTO. La seconde instruction indique que SAS doit utiliser (SET) la table TEST, située actuellement dans la bibliothèque temporaire. Le RUN demande l'exécution du programme.

Dans le cas où votre bibliothèque contient plusieurs chemins physiques, SAS écrira votre table à l'emplacement indiqué par le premier chemin. Dans le cas LAMAR à double chemin (voir programme 1.3), les tables seront écrites dans 'd:\mes documents\ sas exemple'.

Des tables peuvent avoir des noms identiques tant qu'elles sont situées dans des bibliothèques différentes. À l'issue de l'exécution du programme 1.4, vous disposez de deux tables TEST, une dans la bibliothèque WORK, l'autre dans la bibliothèque TOTO.

Il est possible de créer directement la table SAS dans une bibliothèque définie au préalable :

rogramme 1.5

```
ATA toto.test;
  INPUT X1 X2 X3 X4 X5 $;
ARDS;
 5 5 1161 ABÉLIEN
 6 3 1336 ALMÉRIC
 1 5 1499 ANIEL
UN;
```

Exercice 1.1 : *Voici des données que vous allez devoir entrer dans une table SAS :*

```
1 t 2 3 a
4 abc 5 6 b
7 z 8 9 c
10 rt 11 12 d
```

Combien de champs avez-vous ? Combien d'enregistrements ? Rédigez le programme qui créera votre table. Vérifiez ensuite, au moyen d'un PROC PRINT et en éditant votre table dans la fenêtre VIEWTABLE, que votre table est bien à l'image des données présentées ici.

Exercice 1.2 : *Parmi les fichiers qui accompagnent cet ouvrage et que vous avez téléchargés sur* ***www.sas-sr.com*** *puis décompressés à un emplacement sur votre disque dur, il apparaît une table CHAP1.*

1. *Réalisez un PROC PRINT sur cette table.*
2. *Ouvrez-la au moyen de la fenêtre VIEWTABLE.*
3. *Si vous n'arrivez pas à répondre aux questions 1 et 2, essayez d'accéder à la table CHAP1 en passant par l'onglet EXPLORATEUR de SAS puis par « Poste de travail »... Le contenu de la table CHAP1 vous fournira des informations qui vous permettront de trouver les solutions aux deux premières questions.*

1.5 Compréhension approfondie du cas simple

Avant d'aller plus loin dans la création de tables SAS, voyons ce qui se passe dans les cas suivants afin de bien comprendre que ce qui suit votre instruction INPUT conditionne votre résultat – nos différents exemples nous permettront aussi d'introduire la gestion des valeurs manquantes dans une table SAS.

ogramme 1.6

```
TA test;
 INPUT x1 x2 x3 x4 x5;
RDS;
2 3 4 5
7 8 9 10
 12 13 14 15
 17 18 19 20
 22 23 24 25
```

Résultat 1.1

Obs	x1	x2	x3	x4	x5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20
5	21	22	23	24	25

Le programme 1.6 représente le cas standard : par enregistrement, nous avons cinq champs – cinq variables sont déclarées au moyen de l'instruction INPUT.

Programme 1.7

```
DATA test;
   INPUT x1 x2 x3      ;
CARDS;
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
;RUN;
```

Résultat 1.2

Obs	x1	x2	x3
1	1	2	3
2	6	7	8
3	11	12	13
4	16	17	18
5	21	22	23

Dans le programme 1.7, nous indiquons à SAS qu'il a trois variables à saisir : cela fait, SAS passe à l'enregistrement suivant et néglige les champs en fin de ligne.

Programme 1.8

```
DATA test;
   INPUT x1 x2 x3 x4 x5;
CARDS;
1
2
3
4
...
...
...
24
25
;RUN;
```

Résultat 1.3

Obs	x1	x2	x3	x4	x5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20
5	21	22	23	24	25

Dans le programme 1.8, nous avons indiqué à SAS qu'il avait cinq variables à saisir : s'il ne trouve pas l'information dans l'enregistrement, il passe à la ligne suivante et donc à l'enregistrement suivant pour rechercher cette information.

À partir de ces deux exemples, vous devez comprendre ceci : si, dans votre ligne INPUT, vous avez spécifié un nombre excessif ou insuffisant de variables par rapport aux champs contenus dans un enregistrement, vous vous exposez à certains problèmes, comme le montre le programme 1.9.

Programme 1.9

```
DATA test;
   INPUT x1 x2 x3 x4 x5 x6;
CARDS;
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
;RUN;
```

Résultat 1.4

Obs	x1	x2	x3	x4	x5	x6
1	1	2	3	4	5	6
2	11	12	13	14	15	16

Il est impératif que les valeurs manquantes soient marquées par quelque chose. Dans le cadre simple qui est le nôtre actuellement (les champs sont séparés par des espaces), retenez que la notation habituelle des valeurs manquantes est le point (« . »), quel que soit le type de variable (numérique ou caractère) que vous avez à créer.

Programme 1.13

```
DATA test;
   INPUT x1 $ x2 x3 x4 x5;
CARDS;
1 2 3 . 5
6 . 8 9 10
.  12 13 14 15
.  17 18 19 20
21 22 .  24 25
;RUN;
```

Résultat 1.8

Obs	x1	x2	x3	x4	x5
1	1	2	3	.	5
2	6	.	8	9	10
3		12	13	14	15
4		17	18	19	20
5	21	22	.	24	25

La variable X1 est une variable caractère puisque nous avons ajouté le signe $ à la suite de X1 dans l'instruction INPUT. Vous constatez que les points qui apparaissent sur cette variable pour les enregistrements 3 et 4 ont été remplacés par un « vide » (signe des valeurs manquantes des variables caractères). Pour les variables numériques, le point est conservé comme signe de valeur manquante.

Exercice 1.3 : *Dans l'état actuel de vos connaissances, de quel type peuvent être les variables de la table MISS que nous présentons ci-dessous ?*

Obs	Z1	Z2	Z3	Z4	Z5	Z6
1	1	4		10	a	5
2	2	.		11	b	
3	3	.		12	c	.
4	A	.		13	d	7

1.7 Variables numériques et variables caractères

Le premier programme présenté dans ce chapitre (voir section 1.4.1) vous a incidemment montré qu'il existait au sein des tables SAS deux types de variables : les **variables numériques** et les **variables caractères**. L'exercice 1.3 vient d'attirer votre attention sur le fait que les valeurs manquantes étaient saisies différemment en fonction du type de la variable. Dans la dernière section de ce chapitre, nous préciserons les différences profondes qui distinguent les variables numériques des variables caractères.

Les variables numériques ont pour modalités des nombres. Les variables caractères ont pour modalités des chaînes de caractères, qui peuvent comprendre des lettres, des chiffres et des signes spéciaux (en fait, n'importe quel caractère possible). Dans ce type de variable, vous pouvez enregistrer une chaîne comportant jusqu'à 32 767 caractères.

Attention cependant : si une variable ne contient que des chiffres, cela ne signifie pas que cette variable est forcément numérique. Vous pouvez ainsi choisir d'enregistrer des données numériques dans une variable caractère (mais il vous sera par exemple impossible de calculer une moyenne sur une variable caractère, même si les modalités ont l'apparence de chiffres).

Sans spécification particulière, vous pouvez faire comprendre à SAS les données suivantes comme modalités d'une variable numérique :

```
23     23.2     -23     00023     2.3E1     2.3E01     230E-1
```

Si vos données *a priori* numériques présentent une autre forme, vous devrez utiliser des INFORMAT (voir section 2.6), filtres qui expliquent à SAS comment ces données doivent être traitées pour devenir modalités d'une variable numérique au sein d'une table. Ce sera notamment le cas pour les données ayant la forme suivante :

```
12%   21/05/2012     3,14     €125,000.00
```

Si vous tentez de faire « entrer » dans une variable numérique ce type de donnée, vous observerez dans votre JOURNAL un message tel que :

```
317  DATA toto;
318  INPUT x;
319  CARDS;
NOTE: Données incorrectes pour x à la ligne 320 1-3.
REGLE :     ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
320         12%
x=. _ERROR_=1 _N_=1
```

Ce message indique qu'entre les colonnes 1 et 3, vous lirez une donnée que SAS ne peut pas interpréter telle quelle pour la placer dans une variable numérique. La règle ajoutée dans ce message vous permet de voir que dans la donnée « 12% », il y a quelque chose que SAS ne comprend pas : c'est bien entendu le signe %.

Il est extrêmement important, lors de la phase de construction de vos tables, d'attribuer à chaque variable son type exact. Toute donnée numérique doit être enregistrée dans une variable numérique. Dans les variables caractères, vous ne devez enregistrer que du texte.

De même, lorsque vous travaillez avec une table que l'on vous a transmise, vous devez connaître le type de chaque variable. Bien souvent, un simple examen des données de la table ne suffit pas, car les apparences sont généralement trompeuses : vous avez en effet la possibilité d'imposer des FORMAT à vos variables qui vont agir sur leur présentation (voir section 5.3). Voici par exemple une table sur laquelle nous avons demandé l'exécution de la procédure PROC PRINT :

Résultat 1.9

Obs	date	somme	miss	virgule
1	18/03/2010	€1.234,50	A	1234,50
2	19/11/2012	€2.345,60	5	2345,60

Malgré la présence du signe / dans la variable DATE, du signe € (SOMME), d'une virgule (SOMME et VIRGULE) et d'une lettre A (MISS), ces variables sont toutes numériques.

L'examen d'une table à partir de la fenêtre VIEWTABLE peut habituellement vous renseigner. En effet, les modalités des variables caractères y sont alignées à gauche, et celles des variables numériques le sont à droite (voir figure 1.5, section 1.3 pour

une illustration). Pour plus de sûreté, vous pourrez effectuer un PROC CONTENTS sur votre table (voir section 5.1). Voici par exemple un extrait des informations obtenues d'un PROC CONTENTS sur la table que nous avons présentée dans le résultat 1.9 :

Résultat 1.10

```
        Procédure CONTENTS

Liste alphabétique des variables et des attributs

N    Variable    Type     Long.    Format

1    date        Num.        8    DDMMYY10.
3    miss        Num.        8
2    somme       Num.        8    EUROX9.2
4    virgule     Num.        8    NUMX7.2
```

Faites attention à la casse des modalités des variables caractères : si majuscules et minuscules n'ont aucune importance dans les noms des bibliothèques, tables et variables, ce n'est pas du tout le cas pour les modalités elles-mêmes. La modalité « Toto » n'est absolument pas équivalente à « TOTO » ou « toto ».

Concernant également les variables caractères, notez que si vous faites référence dans vos programmes à des modalités particulières, celles-ci doivent nécessairement apparaître entre quotes (simples ou doubles).

```
IF X='A' THEN ...;
```

Cette instruction signifie : si la variable X a pour modalité A, alors…

```
IF X=A THEN ...;
```

Cette instruction signifie : si les variables X et A présentent la même modalité, alors…

Dans ce chapitre, nous avons appris comment SAS fonctionne par défaut lorsqu'il crée une table qui doit contenir des variables numériques et des variables caractères. Nous avons aussi vu qu'au moyen d'options, il était possible de modifier ce fonctionnement par défaut. Nous pouvons donc à présent nous intéresser à des cas moins standard.

2

Aller plus loin dans la création de tables SAS

Au sommaire de ce chapitre :

- 2.1 La création d'une table SAS à partir d'un fichier de données
- 2.2 Autres indicateurs de séparation des champs
- 2.3 Les enregistrements formatés en colonnes
- 2.4 Plusieurs enregistrements pour construire une observation
- 2.5 La gestion des valeurs manquantes
- 2.6 Les INFORMAT
- 2.7 SAS, Excel, l'importation et l'exportation de tables
- 2.8 Le débogage des programmes de création de table

Les données que vous utiliserez pour créer une table SAS n'auront pas toujo la forme indiquée dans le premier chapitre. Ce deuxième chapitre décrit les outils programmation utiles pour traiter des cas s'éloignant du cas simple afin de créer tables SAS. Nous préciserons plus avant l'objet particulier de ce chapitre dan section 2.1.2.

2.1 La création d'une table SAS à partir d'un fichier de données[1]

2.1.1 Principes

Imaginons que vous ayez 10 000 enregistrements à faire entrer dans votre table et que ceux-ci se trouvent dans un fichier au format TXT sur votre disque dur. N traitons ce genre de données comme ci-après.

```
TA test;
 INFILE 'C:\intro_SAS\fichiers\test21.txt';
 INPUT X1 X2 X3 X4 X5 $;
;
```

Dans notre exemple, la structure de l'instruction INPUT indique que les champs dans les enregistrements sont séparés par des espaces. Gardez bien à l'esprit que l'écriture d'un programme incluant le couple d'instructions INFILE/INPUT nécessite une parfaite connaissance de la structure des données à transformer en une table SAS. Si les données ne sont pas à l'écran, nous considérerons dans cette section que leur structure est connue – nous présentons dans la section 2.8 les outils SAS permettant de comprendre l'organisation des données.

Après l'instruction INFILE, vous devez préciser entre quotes le chemin d'accès à votre fichier. Ici, le fichier à traiter est un fichier TXT. SAS peut traiter des fichiers *.DAT ou *.CSV au moyen de l'instruction INFILE[1]. Il peut être utile de limiter dans un premier temps les traitements à un nombre réduit d'enregistrements. Par exemple, vous avez 100 000 enregistrements mais, dans une phase préparatoire, vous souhaitez (pour gagner du temps) traiter uniquement un échantillon : vous pouvez limiter le nombre d'enregistrements à lire au moyen de l'option OBS=. Vous pouvez aussi préciser quel enregistrement considérer dans le fichier pour construire la première observation, grâce à l'option FIRSTOBS=. Faites attention si vous combinez les deux options :

gramme 2.2

```
 test;
NFILE 'C:\intro_SAS\fichiers\test22.txt' obs=50 FIRSTOBS=20;
NPUT X1 X2 X3 X4 X5 $;
```

Votre table n'affichera que les enregistrements représentés sur les lignes 20 à 50 (soit 31 observations). OBS sera donc obligatoirement supérieur à FIRSTOBS. L'option OBS=x indique à SAS qu'il doit lire jusqu'au x[ième] enregistrement. Les options OBS= et FIRSTOBS= vous seront aussi utiles si, par exemple, dans un fichier de 50 enregistrements à transformer en table SAS, des données inutiles occupent les deux premières lignes et la dernière. Vous pourrez exécuter le programme 2.3.

ramme 2.3

```
 test;
NFILE 'C:\intro_SAS\fichiers\test23.txt' FIRSTOBS=3 OBS=49;
NPUT X1 X2 X3 X4 X5 $;
```

Il est possible que chaque enregistrement du fichier soit particulièrement long. Tant que tous les enregistrements contiennent moins de 256 caractères (blancs compris),

I, CSV et DAT sont les formats les plus usuels d'enregistrement des données en ASCII. L'instruction INFILE peut gérer ıtres formats. Consultez l'aide SAS pour plus d'informations.

cela ne pose pas de problème. Si au moins un enregistrement comptait un nombre plus important de caractères, il faudrait l'indiquer à SAS *via* l'option LRECL= dans l'instruction INFILE.

Programme 2.4

```
DATA test;
   INFILE 'C:\intro_SAS\fichiers\test24.txt' LRECL=600;
   INPUT X1-X200;
RUN;
```

Dans le cas présent, 200 variables numériques sont à créer et l'instruction INPUT X1-X200 va créer les variables X1, X2, X3… X200. Elles ont toutes des modalités comprises entre 10 et 99, et un espace sépare chaque champ. Un enregistrement contient donc 200 × 2 + 199 = 599 caractères.

Comment être sûr que SAS a lu l'intégralité de l'enregistrement et comment savoir, sans ouvrir le fichier, que vous n'avez pas besoin de l'option LRECL= ? Lorsque vous créez une table SAS, dans la fenêtre JOURNAL, vous aurez, parmi les messages en bleu, le commentaire suivant :

```
NOTE: 6 enregistrements lus dans infile 'C:\intro_SAS\fichiers\bourvil2.txt'.
      La longueur min. de l'enregistrement était 30.
      La longueur max. de l'enregistrement était 41.
```

SAS vous indique ici que la longueur maximale des enregistrements qu'il a traités est de 41 caractères : vous n'avez pas besoin de l'option LRECL=. S'il devait indiquer 256, cela signifierait pour vous **l'obligation** de spécifier un LRECL suffisamment grand pour pouvoir lire dans la fenêtre JOURNAL une longueur maximum de l'enregistrement inférieure au LRECL spécifié.

Vous disposez aussi d'un module d'importation des données accessible *via* le menu **Fichier**, puis **Importer des données**. Nous traiterons ce module dans la section 2.7.1. Il est particulièrement utile lorsque votre séparateur de champs est la tabulation, comme nous le verrons par la suite.

L'utilisation de l'instruction INFILE oblige pour l'instant à écrire le chemin DOS complet vers le fichier que vous souhaitez voir transformé en une table SAS. Même en utilisant l'explorateur Windows pour copier-coller ce lien dans l'éditeur, l'écriture du chemin DOS peut être fastidieuse si au cours d'une même session, vous devez créer plusieurs tables. L'instruction FILENAME peut simplifier votre travail. Comme l'instruction LIBNAME, FILENAME est une instruction globale : elle agit sur votre environnement et doit donc être exécutée en dehors des étapes DATA ou PROC.

```
FILENAME brut 'C:\intro_SAS\fichiers';
```

L'instruction FILENAME permet de préciser un dossier dans lequel est stocké l'ensemble des fichiers que vous souhaitez transformer en tables SAS. Nous nommons ce dossier « brut ». Lors de la création de la table, vous pouvez directement faire référence à ce dossier au moyen de la programmation suivante :

```
DATA test;
   INFILE brut(test_A.txt);
-;
RUN;
```

```
FILENAME fichier 'C:\intro_SAS\fichiers\test_B.txt';
```

L'étape de création de votre table commencera alors de la manière suivante :

```
DATA test;
   INFILE fichier;
…;
RUN;
```

L'écriture du chemin DOS complet dans l'instruction INFILE n'est pas obligatoire, comme le montre le programme 2.5.

Programme 2.5

```
DATA test;
   INFILE 'test25.txt';
   INPUT X1 X2 X3 X4 X5 $;
RUN;
```

Ce programme peut créer une table si le fichier de données se trouve à l'emplacement par défaut. Cet emplacement est indiqué au bas de votre écran : normalement, à droite (voir figure 2.1), mais il peut apparaître à gauche si vous avez fait disparaître la ligne de message.

Figure 2.1 • Emplacement par défaut.

En double-cliquant sur l'emplacement par défaut, vous pouvez le modifier et indiquer, par exemple, le dossier qui contient vos fichiers de données : ainsi, lorsque vous créerez vos tables, seul le nom du fichier sera nécessaire (vous n'aurez pas besoin de préciser le chemin DOS).

2.1.2 L'INPUT BUFFER

Avant de poursuivre, il est important de comprendre que les données, pour l'instant extérieures à SAS, transitent dans une zone de mémoire appelée INPUT BUFFER avant d'être envoyées dans une seconde zone de mémoire (le PROGRAM DATA VECTOR, ou PDV), puis dans la table que vous créez. Nous précisons dans cette section le fonctionnement de l'INPUT BUFFER.

C'est l'instruction INPUT qui permet le chargement de données extérieures à SAS (*via* CARDS ou INFILE) dans l'INPUT BUFFER. Par défaut, SAS charge **un seul enregistrement à la fois** (de 256 caractères par défaut, puisque LRECL=256).

Il est extrêmement important de comprendre que SAS traite les données « à la volée » : l'exécution de votre programme SAS prend un enregistrement et exécute le programme que vous avez écrit uniquement sur cet enregistrement. **Une fois celui-ci traité, un nouvel enregistrement est chargé et le programme est à nouveau exécuté.**

…fonctionnement de SAS tout aussi important et sur laquelle nous reviendrons principalement dans le chapitre 3 : lorsque SAS trait une observation, par défaut, **il n'a aucune information sur la donnée suivante ni su les données qu'il a déjà traitées.**

Le recours à l'INPUT BUFFER n'intervient qu'au moment de la création d'une tab à partir d'un fichier extérieur à SAS – il n'est pas utilisé lorsque vous créez une table partir d'une autre table où lorsque vous modifiez une table (voir chapitre 3).

Partez de l'idée que les enregistrements présents dans l'INPUT BUFFER sont absolu ment incompréhensibles tels quels. Vous allez devoir donner à SAS un ensemble de clé qui lui permettront de comprendre comment les enregistrements doivent être interpré tés pour être transformés en données SAS. Nous traitons dans les sections qui suiven un certain nombre d'éléments nécessaires à l'interprétation des données pour l'ins tant situées dans l'INPUT BUFFER. Pour interpréter ces données, SAS doit en effet :

- Savoir quel est le séparateur de champs : l'espace (séparateur par défaut) ou autr chose ? Voir section 2.2.
- Savoir où et comment, dans l'INPUT BUFFER, il retrouvera les champs de variables à saisir. Voir section 2.3.
- Comprendre l'organisation des données : un enregistrement pour une observa tion (cas général) ou une autre organisation ? Voir section 2.4.
- Comprendre les signes de valeurs manquantes. Voir section 2.5.
- Savoir s'il doit interpréter tel ou tel champ pour construire la modalité SAS ; cette interprétation passera alors par des INFORMAT. Voir section 2.6.

Ce sont essentiellement les contenus des instructions **INPUT** et **INFILE** et les **INFOR-MAT** qui vont donner à SAS ces différentes clés permettant la transformation des enregistrements de ce fichier entrant (**IN**) en une table SAS. Si vous ne fournissez pas les bonnes clés, SAS ne pourra pas interpréter correctement vos données et votre table ne sera pas l'image exacte de celles-ci.

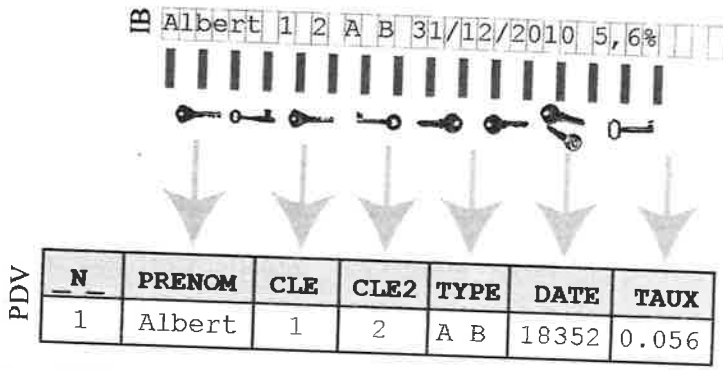


Figure 2.2 • L'INPUT BUFFER.

Une fois interprétés au moyen des différentes clés indiquées, les enregistrements deviendront des modalités de variables SAS. Ces modalités seront placées dans le PDV avant d'être versées dans la table à créer lorsque le programme sera intégralement exécuté sur l'observation courante. Nous reviendrons au début du chapitre 3 sur le fonctionnement du PDV.

2.2 Autres indicateurs de séparation des champs

Nous avons vu que le séparateur de champs par défaut est l'espace, et que le séparateur de champs permet à SAS de savoir quand prend fin une modalité. Vous rencontrerez très certainement des fichiers utilisant un autre séparateur de champs. Des options d'INFILE vous permettront de traiter parfaitement ce type de fichiers. Si vous affichez vos données dans la fenêtre EDITEUR, vous pourrez toujours mobiliser les options d'INFILE *via* l'instruction :

```
INFILE CARDS votre_option;
```

2.2.1 Champs séparés par des tabulations

Imaginons que les champs soient séparés par des tabulations (ce sera le cas dans votre fenêtre EDITEUR si vous copiez-collez d'Excel vers SAS). Si vous passez par une instruction CARDS, cela ne posera aucun problème : votre programmation ne changera pas et la table sera parfaitement créée.

Programme 2.6

```
DATA test;
   INPUT X1 X2 X3 X4 X5 $;
CARDS;
78    5     5     1161  ABÉLIEN
35    6     3     1336  ALMÉRIC
82    1     5     1499  ANIEL
;RUN;
```

En revanche, si vous cherchez à créer une table SAS à partir d'un fichier externe à SAS, il vous faudra utiliser l'instruction INFILE et l'option EXPANDTABS (voir programme 2.7).

Programme 2.7

```
DATA test;
   INFILE 'C:\intro_SAS\fichiers\test27.txt' EXPANDTABS;
   INPUT X1 X2 X3 X4 X5 $;
RUN;
```

Soyez prudent si le séparateur est une tabulation. Vous trouverez parmi les fichiers exemples de cet ouvrage un fichier appelé TEST100.TXT qui comprend 9 enregistrements et 100 champs séparés par des tabulations. Dans le premier enregistrement, ces champs sont tous égaux à 1 ; dans le deuxième, ils sont tous égaux à 2 ; dans le troisième, ils sont tous égaux à 3, etc. Vous pourriez donc considérer que la longueur de chaque enregistrement est égale à 199 (100 caractères et 99 tabulations). Pour demander la création d'une table à partir de ce fichier, vous pourriez écrire le programme 2.8.

Programme 2.8

```
DATA test;
   INFILE 'C:\intro_SAS\fichiers\test100.txt' EXPANDTABS;
   INPUT x1-x100;
RUN;
```

Dans votre fenêtre JOURNAL, vous lirez alors, parmi d'autres informations :

```
NOTE: 9 enregistrements lus dans infile 'C:\intro_SAS\fichiers\test100.txt'.
      La longueur min. de l'enregistrement était 199.
      La longueur max. de l'enregistrement était 199.
```

Puis, immédiatement en dessous :

```
      Une ou plusieurs lignes ont été tronquées.
NOTE: SAS est allé à la ligne lorsque l'instruction INPUT a atteint la fin d'une ligne.
NOTE: La table WORK.TEST a 2 observations et 100 variables.
```

La table n'a pas été correctement créée puisque vous n'avez pas les 9 observations prévues. SAS vous indique qu'il a tronqué une ou plusieurs lignes. Demandez maintenant, au moyen d'un PROC PRINT, l'édition de la table que vous venez de créer. Vous constaterez qu'il y a en effet un problème.

Les remarques reprises dans le JOURNAL sont symptomatiques d'une longueur d'enregistrement supérieure aux 256 caractères prévus par défaut. Il se trouve cependant que SAS vous indique que la longueur maximale de vos enregistrements est égale à 199.

Lorsque la tabulation est utilisée comme séparateur de champs, dans le JOURNAL, une tabulation compte pour un caractère. Cependant, en interne à SAS, une tabulation compte pour 7 caractères. La longueur d'enregistrement est donc non pas de 199 caractères mais de

$100 + 99 \times 7$ caractères = 793 caractères.

En spécifiant LRECL=793 (ou plus), votre table sera parfaitement créée. Si vous indiquez LRECL=792, votre table sera différente de votre fichier de départ et vous aurez toujours dans votre JOURNAL le message suivant :

```
Une ou plusieurs lignes ont été tronquées.
```

Il conviendra donc dans ce cas de ne pas faire confiance à la longueur maximale de l'enregistrement et de donner une valeur suffisamment grande à LRECL pour ne plus obtenir le message repris ci-dessus.

2.2.2 Champs séparés par un caractère quelconque

Si les champs sont séparés par un caractère quelconque (1?2?3?... par exemple), vous utiliserez l'option DLM= et indiquerez entre quotes le caractère utilisé :

Programme 2.9

```
DATA test;
   INFILE 'C:\intro_SAS\fichiers\test29.txt' DLM='?';
   INPUT X1 X2 X3 X4 X5 $;
RUN;
```

Attention cependant à cette option DLM=. Le caractère de séparation que vous choisissez ne doit pas apparaître au sein des modalités des variables que vous voulez créer. La solution qui consiste à opter pour deux caractères rares et à utiliser ensuite DLM= n'est pas bonne, comme le montre le programme 2.10.

Programme 2.10

```
DATA test;
   INFILE CARDS DLM='@#';
   INPUT email $ com $;
CARDS;
a@esa.fr@#A
;RUN;
```

Les enregistrements sont composés d'une adresse e-mail et d'une seconde variable COM. Si vous exécutez ce programme, vous constaterez qu'il ne fonctionne pas comme nous le souhaitions. Un PROC PRINT donne en effet :

```
Obs    email     com
 1       a      esa.fr
```

Lorsque vous indiquez deux caractères, cela signifie en fait que les séparateurs peuvent être l'un des deux caractères : *via* l'option DLM=, vous offrez à SAS une liste des séparateurs de champs possibles. Si votre séparateur de champs est une chaîne d'au moins deux caractères, comme dans le cas du programme 2.10, vous pouvez utiliser l'option DLMSTR= au lieu de DLM=.

Des codes clavier Alt permettent de créer des caractères rares – voir par exemple ce site pour une liste des codes possibles :

www.toutimages.com/codes_caracteres.htm

Pour créer un caractère rare, il suffit de taper le code à quatre chiffres correspondant, tout en maintenant enfoncée la touche Alt du clavier. Par exemple, pour le caractère Þ (le Thorn islandais majuscule), le code est : Alt 0222.

L'utilisation d'un séparateur de champs particulier vous sera particulièrement utile si les modalités de vos variables caractères doivent contenir des espaces et si les enregistrements ne sont pas formatés en colonnes (voir section 2.3).

Si l'espace est le séparateur par défaut et qu'il peut aussi apparaître au sein des modalités, SAS ne pourra pas comprendre la différence entre l'espace au milieu du champ et l'espace qui sépare les champs de deux variables distinctes.

Le programme 2.11 propose un exemple. X2 est numérique et vaut forcément 1 ; le texte avant le 1 est à mettre dans la variable caractère X1.

Programme 2.11

```
DATA test;
   INPUT X1 $ X2;
CARDS;
aa 1
aa a 1
aa a a 1
;RUN;
```

Dans le cas présent, vous ne pourrez pas créer votre table SAS. S'il y a des espaces à l'intérieur des champs, vous devez absolument créer une distinction entre le séparateur de champs et l'espace qui intervient dans la modalité. Si vous ne disposez que d'un fichier TXT[1], c'est manuellement que vous devrez intervenir en introduisant, par exemple, deux espaces entre le dernier « a » et le 1.

Programme 2.12

```
DATA test;
   INPUT X1 & $ X2;
CARDS;
aa  1
aa a  1
aa a a  1
;RUN;
```

Le & placé entre X1 et son INFORMAT $ indique à SAS qu'il doit attendre de voir au moins deux séparateurs de champs consécutifs avant de passer à l'enregistrement de la variable suivante. Vous avez aussi la possibilité d'introduire un autre séparateur de champs, lequel conduira SAS à considérer l'espace comme un caractère quelconque :

Programme 2.13

```
DATA test;
   INFILE CARDS DLM='#';
   INPUT X1 $ X2;
CARDS;
aa#1
aa a#1
aa a a#1
;RUN;
```

En revanche, l'utilisation d'une tabulation comme séparateur de champs ne fonctionnera pas. Dans le fichier TABU.TXT, qui reprend les données ci-dessus, le dièse a été remplacé par une tabulation. Pour créer une table, *a priori*, nous exécuterions le programme 2.14.

Programme 2.14

```
DATA tabu;
   INFILE 'C:\intro_SAS\fichiers\tabu.txt' EXPANDTABS;
   INPUT X1 $ X2;
RUN;
```

Or ce programme ne fonctionne pas. L'option EXPANDTABS continue à considérer l'espace comme un séparateur de champs. Votre table ressemblera au résultat 2.1.

Résultat 2.1

Obs	X1	X2
1	aa	1
2	aa	.
3	aa	.

Vous devrez ouvrir votre fichier avec un éditeur de texte quelconque ou *via* FSLIST (voir section 2.8) et remplacer votre tabulation par un caractère qui vous permettra d'utiliser l'option DLM=. Plus simplement, vous pouvez ouvrir le fichier dans Excel puis l'enregistrer au format XLS, et ensuite utiliser le module d'importation (voir section 2.7.1). Il est aussi possible d'ouvrir directement le classeur Excel à partir de SAS (voir section 2.7.2).

2.2.3 Champs séparés par un point-virgule

Si vos données sont dans un fichier, une instruction INFILE accompagnée de l'option DLM=';' suffira. (Vous pouvez essayer avec le fichier d'accompagnement POINTVIRGULE.TXT.) S'il vous faut inclure les données dans votre programme, vous devrez utiliser conjointement DLM= et l'instruction CARDS4 ou DATALINES4.

Programme 2.15

```
DATA test;
   INFILE CARDS DLM=';';
   INPUT x1 x2 x3 x4 x5;
CARDS4;
1;2;3;4;5
1;2;3;4;5
;;;;
RUN;
```

Insérez ';;;;' (après un retour chariot) avant votre RUN habituel afin d'indiquer à SAS que la plage de données à saisir est terminée.

© 2.3 Les enregistrements formatés en colonnes

Deux principaux formats d'organisation des enregistrements existent : les enregistre-

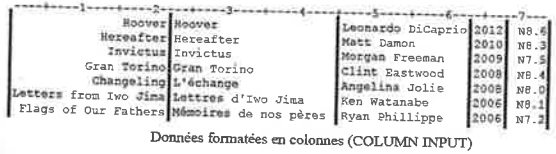
présentent plusieurs avantages, le premier d'entre eux étant la possibilité de saisir des modalités caractères contenant des espaces, puisque vous n'avez plus besoin de séparateurs de champs.

2.3.1 Cas général

Les enregistrements présentés en colonnes ont généralement la forme suivante :

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7---
            Hoover Hoover                    Leonardo DiCaprio  2012  N8.6
         Hereafter Hereafter                 Matt Damon         2010  N8.3
          Invictus Invictus                  Morgan Freeman     2009  N7.5
      Gran Torino Gran Torino                Clint Eastwood     2008  N8.4
        Changeling L'échange                 Angelina Jolie     2008  N8.0
Letters from Iwo Jima Lettres d'Iwo Jima     Ken Watanabe       2006  N8.1
Flags of Our Fathers Mémoires de nos pères   Ryan Phillippe     2006  N7.2
```

La règle ne fait pas partie des données mais permet de lire les positions des divers champs. Les données apparaissent donc ici en colonnes : vous avez la possibilité de tracer des lignes verticales qui vont parfaitement séparer les différents champs que vous avez à saisir. La figure 2.3 illustre la différence entre des données COLUMN INPUT et des données LIST INPUT.



Données formatées en colonnes (COLUMN INPUT)

```
Hoover Hoover Leonardo DiCaprio 2012 N8.6
Hereafter Hereafter Matt Damon 2010 N8.3
Invictus Invictus Morgan Freeman 2009 N7.5
Gran Torino Gran Torino Clint Eastwood 2008 N8.4
Changeling L'échange Angelina Jolie 2008 N8.0
Letters from Iwo Jima Lettres d'Iwo Jima Ken Watanabe 2006 N8.1
Flags of Our Fathers Mémoires de nos pères Ryan Phillippe 2006 N7.2
```

Données non formatées en colonnes (LIST INPUT)

Figure 2.3 • Données COLUMN INPUT et LIST INPUT.

Les données avec lesquelles nous allons construire une table SAS sont relatives aux sept films réalisés par Clint Eastwood entre 2006 et 2012. Le titre original occupe les colonnes 1 à 21 ; le titre français, les colonnes 23 à 43 ; le nom de l'acteur principal, les colonnes 46 à 62 ; et l'année de sortie du film, les colonnes 64 à 67. Enfin, une note sur 10, précédée d'un 'N', se trouve entre les colonnes 71 et 73[1].

Si vous avez saisi les données avec CARDS, une petite fenêtre dans le coin inférieur droit de votre fenêtre EDITEUR vous indique la ligne (Ln) sur laquelle vous vous trouvez, ainsi que le numéro de la colonne (Col) qui suit votre curseur (voir figure 2.1). Vous pouvez ainsi repérer facilement les colonnes qui contiennent vos modalités. Si vos données sont présentes dans un fichier extérieur, vous avez la possibilité de recourir aux outils FSLIST et NOTEPAD, présentés dans la section 2.8.

Nous ne pouvons pas créer de table SAS avec les outils vus jusqu'à maintenant. En effet, il faudrait saisir des champs particuliers qui contiennent des espaces ; or, jusqu'ici, l'espace est considéré comme séparateur de champs. Essayons tout de même…

Programme 2.16

```
DATA clint;
INPUT titre_us $ titre_fra $ acteur $ annee note;
CARDS;
*** les données ***
;RUN;
```

Le résultat 2.2 reprend les résultats obtenus du PROC PRINT.

Résultat 2.2

Obs	titre_us	titre_ fra	acteur	annee	note
1	Hoover	Hoover	Leonardo	.	2012
2	Hereafte	Hereafte	Matt	.	2010
3	Invictus	Invictus	Morgan	.	2009
4	Gran	Torino	Gran	.	.
5	Changeli	L'échang	Angelina	.	2008
6	Letters	from	Iwo	.	.
7	Flags	of	Our	.	.

L'instruction INPUT indique à SAS que les champs sont séparés par des espaces. Il enregistre donc, dans la variable TITRE_US, ce qu'il voit, jusqu'à ce qu'il rencontre un espace ; il enregistre pour TITRE_FRA le deuxième mot de la ligne, puis le mot suivant en tant que modalité pour la variable ACTEUR, etc.

Puisque les données sont ici présentées en colonnes, nous allons adapter notre programmation et spécifier les emplacements colonnes des variables[1]. La ligne INPUT devient ainsi :

```
INPUT titre_us $ 1-21 titre_fra $ 23-43 acteur $ 46-62 annee note 71-73;
```

Et votre table SAS sera parfaitement créée. Il n'est pas utile de préciser dans la ligne INPUT l'emplacement de la variable ANNEE, puisque les champs correspondant à cette variable ne comprennent pas de blancs. Lorsque SAS a terminé de saisir la variable ACTEUR, il passe systématiquement les blancs qui peuvent suivre ; il n'entame la saisie de la variable ANNEE que lorsque celle-ci commence effectivement. Des données présentées en colonnes ne vous obligent donc pas à préciser systématiquement les emplacements colonnes de chaque champ. Nous spécifions l'emplacement de NOTE pour éviter le 'N' qui précède les notes. C'est ici obligatoire si l'on souhaite enregistrer ces notes dans une variable numérique.

Lorsque vous réalisez un PROC PRINT, les variables s'affichent suivant l'ordre dans lequel elles sont créées (et donc dans lequel elles apparaissent dans l'instruction INPUT). Si vous souhaitez que vos variables soient saisies selon un ordre différent, cela ne pose pas de réel problème tant que vos données sont de type COLUMN INPUT.

La ligne d'INPUT suivante organise les variables dans votre table selon un ordre qui vous sera peut-être plus utile :

```
INPUT annee 64-67 titre_us $ 1-21 titre_fra $ 23-43 note 71-73 acteur $ 46-62;
```

Exercice 2.1 : Créez deux tables SAS à l'aide des données ci-dessous (fichier EXO21.TXT) pour obtenir les tables ci-contre :

	Obs	x1	x2	x3	x4	Obs	Z1	Z2	Z3	Z4	Z5
123456789abc	1	123	45	6789	abc	1	abc	3456	12	9a	8
987654321def	2	987	65	4321	def	2	def	7654	98	1d	2
253698741ghi	3	253	69	8741	ghi	3	ghi	3698	25	1g	4
879654321jkl	4	879	65	4321	jkl	4	jkl	9654	87	1j	2
324578961lmn	5	324	57	8961	lmn	5	lmn	4578	32	11	6
524169387opq	6	524	16	9387	opq	6	opq	4169	52	7o	8

2.3.2 Le pointeur +X

Pour indiquer à SAS la position de vos champs, vous pouvez soit préciser les colonnes qui contiennent les données (positionnement absolu), soit utiliser le pointeur +X (positionnement relatif) :

```
INPUT titre_us $ 1-21 +1 titre_fra $ 23-43 +2 acteur $ 46-62 +1 annee +2 note;
```

Rappel de l'organisation de l'enregistrement :

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7---
           Hoover Hoover                    Leonardo DiCaprio  2012  N8.6
```

Le +1 qui apparaît après l'indication de position 1-21 est un **pointeur**. Il permet un déplacement dans l'enregistrement présent dans l'INPUT BUFFER. Nous demandons à SAS de lire entre les colonnes 1 et 21 la modalité de TITRE_US, puis de se déplacer d'une colonne pour débuter l'enregistrement de la variable suivante (TITRE_FRA).

En réalité, il n'est pas nécessaire de recourir au pointeur +X pour les données utilisées ici, mais celles-ci vous permettront de comprendre la « dangerosité » du +X. Concentrons-nous sur la saisie des variables ACTEUR, ANNEE et NOTE :

```
INPUT … acteur $ 46-62 +1 annee +2 note;
```

Pour saisir la variable ACTEUR, vous précisez à SAS que le champ est placé entre les colonnes 46 et 62 (incluses). SAS n'a pas besoin d'un séparateur de champs (l'espace par défaut) pour clore cette saisie. +1 demande un déplacement d'une colonne (SAS passe donc la colonne 63) et ANNEE peut être lue.

Aucune notation colonne n'est précisée pour ANNEE. SAS a donc besoin de l'espace (séparateur de champs) qui suit pour clore sa saisie. Trois colonnes apparaissent entre ANNEE et NOTE (deux espaces et la lettre N), mais SAS ne doit se déplacer que de deux colonnes pour saisir NOTE.

Maintenant, si vous précisez que la variable ANNEE est à lire entre les colonnes 64 et 67 (incluses), il n'est plus besoin de l'espace qui suit pour comprendre que la saisie de la modalité est terminée : vous devez indiquer +3 pour vous positionner au début du champ correspondant à NOTE.

En ce qui concerne donc les deux dernières variables, les instructions INPUT suivantes sont donc parfaitement équivalentes :

```
INPUT ..... annee +2 note;
INPUT ..... annee 64-67 +3 note;
```

Si vous n'y prenez pas garde, et insérez +3 dans la première instruction ci-dessus, la saisie de NOTE ne commencera qu'à la 72e colonne : vous perdrez la partie entière de la note.

Par conséquent, si vous souhaitez utiliser les possibilités offertes par le pointeur +X au sein de l'instruction INPUT, vous devrez toujours vous demander si SAS a besoin de l'espace qui suit le champ pour clore la saisie de ce champ et pourrez déterminer le nombre de colonnes à sauter.

Enfin, rappelons que l'utilisation des +X ne se justifie pas ici. Cette procédure présente un réel intérêt quand, par exemple, une partie de l'enregistrement est à négliger et qu'elle est d'une longueur constante.

Exercice 2.2 : *Au moyen des données ci-dessous (fichier AZERTY.TXT), créez une table SAS identique à la table présentée à droite :*

```
a A ERTY IOP 12
aa AZE TYUIOP 13
aaa AZERTYUIOP 14
aaaa AZ RT UI P 15
```

Obs	X1	X2
1	a	12
2	aa	13
3	aaa	14
4	aaaa	15

2.3.3 Un cas particulier : le dernier champ est de longueur variable

Imaginons que notre fichier de départ ait cette forme particulière :

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----
Hoover                  Hoover                  2012  8.6  Leonardo DiCaprio
Hereafter               Hereafter               2010  8.3  Matt Damon
Invictus                Invictus                2009  7.5  Morgan Freeman
Gran Torino             Gran Torino             2008  8.4  Clint Eastwood
Changeling              L'échange               2008  8.0  Angelina Jolie
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
```

Vous pourriez penser qu'il n'y a aucune différence, et que nous devrions pouvoir recourir au même type de programmation. Cela ne fonctionne pas comme vous le montre l'exercice 2.3.

Exercice 2.3 :

1. *Transformez les données ci-dessus en table SAS – données sur votre écran : vous passez donc par un CARDS.*
2. *Adaptez votre programme en retirant CARDS et les lignes de données. Vous passez donc par un INFILE. (Fichier à utiliser ici : CLINT2.TXT.)*

À l'issue de cet exercice, vous devez constater que le premier point ne pose aucun problème. En revanche, la création de la table au moyen du fichier TXT ne se passe pas bien. En fait, lorsque vous passez par un INFILE, si le dernier champ de chaque enregistrement est de longueur variable (c'est bien le cas ici, puisque les titres n'ont pas tous le même nombre de caractères), sans l'option TRUNCOVER d'INFILE, SAS pourra forcer la lecture de données présentes dans l'enregistrement suivant.

Pour définir les colonnes entre lesquelles vous devez lire le nom de l'acteur principal, vous allez prendre le nom le plus long et constater qu'il occupe les colonnes 56 à 72. Dans votre fichier, chaque modalité d'ACTEUR ne prend pas forcément 17 caractères (72-56+1). SAS, pour remplir cette variable, va alors chercher le premier bloc de 17 caractères présent dans un même enregistrement et peut passer pour cela à l'enregistrement suivant. Cela vous explique pourquoi, lorsque vous demandez un PROC PRINT, vous observez le résultat 2.3.

Résultat 2.3

Obs	titre_us	titre_fra	annee	note	acteur
1	Hoover	Hoover	2012	8.6	Leonardo DiCaprio
2	Hereafter	Hereafter	2010	8.3	Invictus
3	Gran Torino	Gran Torino	2008	8.4	Changeling
4	Letters from Iwo Jima	Lettres d'Iwo Jima	2006	8.1	Flags of Our Fath

L'option TRUNCOVER d'INFILE indique à SAS qu'il ne doit pas rechercher dans l'enregistrement suivant si on lui demande de lire un certain nombre de caractères pour une variable et qu'il ne trouve pas exactement ce nombre de caractères. Attention, si l'option TRUNCOVER peut souvent vous être utile lors d'un passage par une instruction INFILE pour créer une table, elle vous sera parfaitement inutile lorsque vous passerez par un CARDS : SAS ne force pas la lecture sur l'enregistrement suivant s'il lui manque des caractères lors d'une entrée de données par CARDS, SAS ne force la lecture sur l'enregistrement suivant que s'il ne rencontre **aucun** caractère (voir section 1.5).

2.3.4 Le pointeur @X

Lorsque les enregistrements sont présentés en colonnes, il existe une seconde manière de se déplacer à l'intérieur d'un entregistrement. Ainsi :

```
INPUT @64 annee titre_us $ 1-21  titre_fra $ 23-43 acteur $ 46-62 @71 note;
```

Le pointeur @X indique à SAS qu'il doit se positionner en x^ième colonne pour lire la variable dont le nom suit. Cette instruction évite les calculs nécessaires à l'utilisation du pointeur +X pour passer d'un champ à un autre ou les notations de colonnes si vous souhaitez, comme ici, enregistrer vos variables dans un ordre différent de celui de vos données. Vous devez uniquement connaître la position à laquelle le champ commence.

Les données sur lesquelles nous nous appuyons dans ce chapitre ne nécessitent pas absolument le recours aux pointeurs @X. Un pointeur @X permet notamment d'écarter des champs lors de la construction de la table. Cela dit, l'utilisation seule de @X ne règle pas les cas pour lesquels un espace doit apparaître dans la modalité que vous souhaitez saisir, si l'espace est aussi séparateur de champs. Enfin, comme le montre notre exemple, ce type de programmation permet aussi de « revenir en arrière » et d'effectuer un second passage sur l'enregistrement[1].

1. Il est aussi possible de revenir « en arrière » avec le pointeur +X : –X n'existe pas mais +(–X) est possible.

2.3.5 Une seconde application de @X : @'texte'

Imaginons que nous disposions des données suivantes :

```
)m: Sophie mljfpé"nv,)àé"'j,mlcé")à fié"'jh)ç pojé' remarque: BB
: fpi'fp   prénom: Hélène  remarque: A
)rénom: Julie ,^éap,n ofpihé   remarque: CCC
 prénom: Marie jpoujpojerf rj,opj  remarque: AB
 rfopj fpojp^zed é")j,prénom: Erwan jjzperfhn npzepîkremarque: AC
'énom: Emilie jpojef' feiohefp   remarque: DDD
)om: Valérie remarque: CC f,pzeiojfpoze fepifhpzef
)z foeifhpi poopop   prénom: Sylvain morf5rf5zeffoi  remarque: AC
'énom: Marie remarque: BB  efiherfioe' foé"'pbh
'fiophr p^çj" dfprénom: Claire remarque: A pou")é"'n )')uhnç)uç'h
```

Un bug informatique a créé ce fichier dans lequel apparaissent sur chaque ligne un prénom et une remarque. Vous pouvez malgré tout en faire une table SAS en profitant des récurrences 'prénom:' et 'remarque:'.

ramme 2.17

```
 test;
NPUT @'prénom:' eleve :$10. @'remarque:' attitude $;
5;

 lonnées
;
```

Nous avons ici une seconde application du pointeur @ dans une instruction INPUT. Nous avons vu l'intérêt d'un @5 X1 qui demandait à SAS de se placer en cinquième colonne pour lire la variable X1. @'prénom:' ELEVE demande à SAS d'avancer dans l'enregistrement et de trouver 'prenom:' pour ensuite lire la variable ELEVE (:$10. est un INFORMAT – voir section 2.6).

Exercice 2.4 : *Essayez cette manœuvre dans le fichier ABCD.TXT. Est-ce que le nombre d'enregistrements du fichier est bien égal au nombre d'observations de votre table ? Modifiez votre programmation de façon que votre table soit fidèle à votre fichier de départ.*

Exercice 2.5 : *@X vous sera particulièrement utile lorsque vous construirez des tables au moyen de données contenues dans des pages HTML, par exemple. Vous trouverez à l'adresse suivante une feuille HTML contenant des données que l'on voudrait bien utiliser pour construire une table.*

http://www.sas-sr.com/files/bluenote1.html

Nous avons édité le code source de cette page et l'avons enregistré dans un fichier texte BLUENOTE.TXT. Les données que nous souhaitons voir s'afficher dans une table SAS ressemblent à ceci :

```
BLP 4206  <a href="../#blp-4206" name="blp-4206">Sam Rivers - Contours</a>
```

Vous n'avez pas à rédiger de programme (pour l'instant… voir exercice 2.15, section 2.6.2) destiné à construire une table. Vous devez simplement examiner ce fichier afin de comprendre comment sont organisées les données et voir quels éléments vont permettre à SAS de repérer la référence du disque (BLP 4206 ou blp-4206), le nom de l'artiste (Sam Rivers) et le nom de l'album (Contours).

2.4 Plusieurs enregistrements pour construire une observation

Dans toutes les tables construites jusqu'à maintenant, chaque observation a toujours été obtenue au moyen d'un enregistrement unique (une ligne de données). Nous allons voir dans cette section des cas dans lesquels plusieurs observations peuvent être construites au moyen d'un même enregistrement, et des cas dans lesquels il faudra combiner plusieurs enregistrements pour construire une observation.

2.4.1 Un enregistrement, plusieurs observations : @@ et @

Exercice 2.6 : *Dans le fichier FLORIDE.TXT, vous disposez des précipitations observées en Floride depuis plus d'un siècle dans les vingt-deux stations d'observations de cet État. Malheureusement, nous avons fait une mauvaise manipulation en préparant ce fichier : les champs sont bien séparés d'un espace, mais il n'y a aucun saut de ligne.*

Exemple : 80211 1903 -9999 80211 1904 2507 80211 1905 4876 80211 1906 5023 80211 1907 6356…

Sur cette ligne, 80211 est le numéro de la base ; 1903, l'année d'observation ; –9999 désigne la valeur manquante ; sinon, ce sont les précipitations annuelles que vous observerez.

Créez une table SAS avec ces données. Il y a 2 299 observations. L'enregistrement que vous avez à traiter comprend un peu moins de 38 000 caractères.

Vous ne disposez pas encore des outils permettant d'apporter des réponses pertinentes aux questions posées dans cet exercice – l'important, ici, est de comprendre pourquoi le programme que vous allez rédiger pour répondre à la question posée ne fonctionne pas.

Nous avons vu que, par défaut, lorsque SAS analyse un enregistrement, c'est ce qui suit l'instruction INPUT et les options qui peuvent apparaître dans l'instruction INFILE qui déterminent sa façon de lire les différents champs. Parmi les éléments du fonctionnement par défaut de SAS, nous avons ainsi vu que lorsque SAS a donné à toutes les variables définies par l'instruction INPUT une modalité, il cesse de lire l'enregistrement et passe à l'enregistrement suivant. Dans le cas de l'exercice 2.6, nous ne voulons pas que SAS passe à l'enregistrement suivant (qui n'existe pas, en fait). Cela se fera au moyen de l'option @@ de l'instruction INPUT. Le programme 2.18 traitera correctement le fichier FLORIDE.TXT.

Programme 2.18

```
DATA floride;
   INFILE 'C:\intro_SAS\fichiers\floride.txt' LRECL=38000;
   INPUT base annee pluie @@;
   IF pluie=-9999 THEN pluie=.;
RUN;
```

Puisque la longueur de l'enregistrement dépasse 256 caractères, vous devez spécifier un LRECL suffisamment important.

Exercice 2.7 : *Vous trouverez dans le fichier ORLEANS.TXT les températures journalières maximales observées entre le 1er janvier 2005 et fin juillet 2009 sur Orléans[1]. Les données sont présentées sur 168 lignes mais chaque ligne présente 10 données de températures. Chaque*

1. Source : Klein Tank, A. M. G. *et al.*, 2002, "Daily dataset of 20th-century surface air temperature and precipitation series for the European Climate Assessment", *Int. J. of Climatol.*, 22, 1441-1453. http://eca.knmi.nl