

2

Concepts et vocabulaire pour programmer en SAS

Objectif

Ce chapitre définit les concepts qu'il est essentiel d'avoir intégrés avant de commencer la programmation dans SAS. En particulier, l'organisation des données et la logique du langage. Vous devez lire ce chapitre si vous ne savez pas ce que représente exactement, dans SAS :

- une bibliothèque;
- une table;
- une observation;
- une variable;
- un label;
- un format;
- une étape Data;
- une procédure;
- l'*Output Delivery System* (ODS)...

La première partie de ces explications concerne la manière dont SAS stocke et accède aux données. Quelques instructions et manipulations nécessaires y seront décrites : allouer une bibliothèque, visualiser son contenu, voir une table, voir les variables d'une table. La deuxième partie concerne la programmation, et fait

la distinction entre l'étape Data et les procédures, qui sont les deux grandes étapes de programmation de SAS. Les autres grandes familles d'instructions – options, titres, ODS – seront également présentées à cette occasion. La troisième partie traite du décalage qui existe – à travers les labels et les formats – entre les données vues à l'écran et la réalité qu'il faut programmer. Le chapitre s'achève avec une proposition de schéma de travail pour qui doit programmer avec SAS.

2.1 L'ORGANISATION DES DONNÉES

Le but premier de SAS est la manipulation de données, avant même de commencer les traitements statistiques proprement dits. Là où d'autres logiciels comme Excel se contentent de demander à l'utilisateur où aller chercher les données, SAS demande de lui définir au préalable le ou les emplacements où aller lire et écrire des données avec le système des *bibliothèques*.

Les fichiers de données lus et créés par SAS sont d'un type qui est spécifique à ce logiciel : ce sont des *tables* et des *vues*. SAS stocke également de l'information dans d'autres objets que nous décrirons sommairement. Enfin, les fichiers à importer, s'ils sont plus simples à manipuler dans les versions les plus récentes du logiciel, peuvent toujours nécessiter une déclaration préalable avec un *fileref*.

2.1.1 Les bibliothèques

Définition

Une bibliothèque¹ est un *nom logique*, un raccourci, attribué dans SAS à un répertoire² contenant ou pouvant contenir des données SAS. Il s'agit d'une convention purement interne, destinée à présenter des fichiers sans faire à chaque fois mention de leur emplacement physique.

Par exemple, un fichier SAS appelé VENTES qui se trouverait dans le répertoire « *c:\mes documents\données\SAS* » ne sera pas cité dans un programme par cette arborescence. On aura effectué une correspondance entre ce répertoire et un nom auquel il sera ensuite fait référence dans toute la session SAS; par exemple, DONNEES.

1. Bibliothèque est la traduction du mot anglais *library*, utilisé par SAS. Certains utilisateurs francophones, voulant rester plus fidèles à la phonétique qu'au sens du mot, parlent de « librairie ».

2. L'association d'un répertoire à une bibliothèque n'est que le cas le plus courant. Sur certains types d'ordinateurs (*mainframe* par exemple), une bibliothèque correspond bien à un espace physique – qui s'appelle un PDS et non un répertoire. La suite de cette section montrera également qu'une bibliothèque peut être associée à d'autres réalités : classeur Excel, base Oracle...

Le fichier en question sera alors désigné par un nom en deux parties, DONNEES.VENTES; la première partie est le nom de la bibliothèque dans laquelle le fichier est stocké; la seconde partie est le nom de ce fichier.

Une fois définie, une bibliothèque n'est *allouée* que *pour la durée de la session* SAS. Il est possible cependant de la rendre pseudo-permanente en effectuant son allocation dans un des fichiers exécutés par SAS au démarrage d'une nouvelle session : programme (autoexec.sas) et fichier de configuration. L'interface d'allocation d'une bibliothèque (fin du paragraphe 2.1.1.) permet également, par un troisième moyen, de réallouer une bibliothèque au démarrage de SAS.

Le nom d'une bibliothèque doit satisfaire à plusieurs contraintes :

- ne pas reprendre le nom d'une bibliothèque allouée automatiquement par SAS (WORK, SASUSER, SASHELP et MAPS) ;
- ne pas dépasser 8 caractères;
- ne doit comporter que des lettres non accentuées, des chiffres (mais pas en 1^{re} position dans le nom) et le signe « _ » (underscore ou « blanc souligné »).

Initialement, une bibliothèque correspondait dans SAS à un espace sur le disque pouvant contenir des données SAS. La syntaxe de base, donnée au paragraphe suivant, a donc été longtemps largement suffisante. La multiplicité des contextes (types de plates-formes, client/serveur) et la possibilité d'allouer des « pseudo-bibliothèques » vers des données d'autres applications (Oracle, Excel, etc.) ont considérablement enrichi cette syntaxe, dont nous allons étudier les versions les plus courantes.

Allouer une bibliothèque

La manière la plus courante d'allouer une bibliothèque est d'utiliser l'instruction LIBNAME. Il est préférable d'utiliser cette instruction *en tête de programme*, de manière à la retrouver facilement.

```
| LIBNAME nomBib "chemin";
```

Dans cette syntaxe, nomBib désigne le nom de la bibliothèque, respectant les conventions énumérées au paragraphe précédent. Le chemin est le chemin physique des données. Il doit respecter les conventions du système d'exploitation.

Sous Windows : LIBNAME donnees "c:\mes documents\données\SAS";

Sous Unix : LIBNAME donnees "/commun/donnees/SAS";

Une fois que l'instruction LIBNAME a été correctement exécutée, le message suivant s'affiche dans la fenêtre *Log* :

Library *nomBib* was successfully assigned as follows :

Suivent les indications sur les caractéristiques de cette bibliothèque.

Données de diverses versions de SAS

Si le répertoire spécifié dans le chemin contient des fichiers provenant de différentes versions de SAS, la bibliothèque telle que définie par la syntaxe précédente ne permet d'accéder qu'aux fichiers créés par la version *la plus récente* de SAS.

Pour spécifier que les données à accéder sont celles créées par une version particulière de SAS, la syntaxe devient :

```
LIBNAME nomBib version "chemin";
```

Version est à remplacer par **V6**, **V8** ou **V9** selon les besoins.

Exemple 2.1 – LIBNAME (1) : allocation d'une bibliothèque vers un répertoire

```
LIBNAME donnees6 v6 "c:\documents\sas\donnees";
```

Bibliothèques SAS/ACCESS

À partir de la version 8 de SAS, une instruction LIBNAME permet de définir une pseudo-bibliothèque qui pointe, non pas vers un répertoire, mais vers plusieurs fichiers d'autres applications. Par exemple, ces ensembles de fichiers peuvent être un classeur Excel (chaque feuille sera vue comme un fichier de données séparé) ou une base de données relationnelle comme Microsoft Access, Oracle, SQL Server, DB2, etc. (chaque table sera vue comme un fichier séparé).

L'utilisation de ces instructions LIBNAME nécessite d'avoir une licence SAS/ACCESS TO PC FILES pour Excel et Microsoft Access, SAS/ACCESS TO ORACLE pour une base de données Oracle.

Bibliothèque vers un classeur Excel

```
LIBNAME nomBib EXCEL "chemin et nom du fichier.xls"  
VERSION = versionExcel HEADER = YES|NO;
```

Dans cette instruction (disponible à partir de la version 9 de SAS) les deux options VERSION et HEADER sont spécifiques à la création d'une pseudo-bibliothèque vers un classeur Excel ou une base Microsoft Access. VERSION peut prendre les valeurs 2002, 2000, 97, 95 ou 5. HEADER permet de signaler si la première ligne des feuilles Excel du classeur contient les noms des colonnes; si c'est le cas, HEADER vaut YES; sinon, HEADER vaut NO.

Exemple 2.2 – LIBNAME (2) : allocation d'une bibliothèque vers un classeur Excel

```
LIBNAME x1Ventes EXCEL "c:\mes documents\donnees\excel\ventes.xls"  
VERSION = 97 HEADER = YES;
```

Bibliothèque vers une base de données relationnelle Oracle

```
LIBNAME nomBib ORACLE USER = identifiant PASSWORD = motDePasse
PATH = "instanceOracle" < SCHEMA = groupe >;
```

Les options USER, PASSWORD et PATH sont spécifiques à la création d'une pseudo-bibliothèque vers une base Oracle (à partir de la version 8 de SAS); elles permettent de désigner respectivement l'identifiant de l'utilisateur et son mot de passe (nécessaires pour se connecter à la base), ainsi que l'instance Oracle (ensemble de tables) à laquelle on souhaite accéder. L'option facultative SCHEMA permet de désigner un groupe de tables Oracle.

Exemple 2.3 – LIBNAME (3) : allocation d'une bibliothèque vers une base Oracle

```
LIBNAME bdVentes ORACLE USER = sas01 PASSWORD = soleil
PATH = "DWPROO1";
```

Il est également possible de demander les informations sensibles (nom d'utilisateur et mot de passe) à travers une fenêtre de dialogue. La syntaxe est alors :

```
LIBNAME nomBib ORACLE DBPROMPT = YES DEFER = NO
PATH = "instanceOracle";
```

L'option DBPROMPT = YES demande cette fenêtre de dialogue, tandis que DEFER = NO permet de ne pas redemander ces informations à chaque utilisation d'une donnée de la bibliothèque.

Bibliothèques sur les machines mainframe

Dans les environnements de type *mainframe*, la question majeure qui se pose à l'allocation d'une bibliothèque concerne les droits sur son contenu. Si on désire partager le contenu de la bibliothèque avec d'autres utilisateurs, chacun n'ayant qu'un droit de lecture sur le contenu, alors il faudra préciser l'option DISP = SHR.

Si on désire écrire dans la bibliothèque (pour peu que des droits soient alloués à l'utilisateur dans ce sens), l'option sera DISP = OLD.

Ces options suivent le nom physique du contenant des données, comme dans l'exemple suivant :

```
LIBNAME donnees "USER.SASDATA.DONNEES" DISP = SHR;
```

Si une bibliothèque n'est pas déclarée en lecture avec DISP = SHR, alors il est bon de la désallouer dès qu'on a fini de se servir de son contenu. En effet, lors de l'allocation avec DISP = OLD, seul l'utilisateur ayant défini cette bibliothèque a le droit d'utiliser son contenu. Pour désallouer une bibliothèque, utiliser l'instruction :

```
LIBNAME nomBib CLEAR;
```

Les bibliothèques en client/serveur : le RLS

Dans le cas d'un fonctionnement client/serveur de SAS, il est possible d'allouer (par une des syntaxes précédentes) une bibliothèque sur le serveur, à l'intérieur d'un bloc RSUBMIT/ENDRSUBMIT. Pour voir le contenu de cette bibliothèque sur le client (PC), il faut alors déclarer la bibliothèque sur la session SAS locale avec :

```
LIBNAME nomBibLoc SLIBREF = nomBibServ SERVER = nomServ;
```

Dans cette syntaxe, *nomBibLoc* est le nom donné dans la session locale à la bibliothèque; *nomBibServ* est le nom donné lors du LIBNAME sur le serveur; *nomServ* est le nom de ce serveur (donné dans la fenêtre *Log* lors de l'exécution de l'instruction RSUBMIT : le message est alors « Remote submit to *nomServ* commencing »).

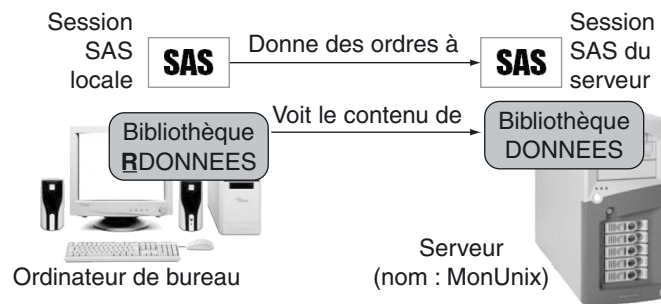


Figure 2.1 — Exemple d'architecture client/serveur

Exemple 2.4 – LIBNAME (4) : allocation d'une bibliothèque sur le serveur et allocation sur le PC d'un lien vers cette bibliothèque (dans la configuration de la figure 2.1)

```
RSUBMIT;
LIBNAME donnees "~/commun/donnees/SAS";
ENDRSUBMIT;
LIBNAME rdonnees SLIBREF = donnees SERVER = MonUnix;
```

Ce système porte le nom de *Remote Library Service* ou RLS. Son utilisation pour visualiser régulièrement des données est très déconseillée, car elle suppose de nombreux transferts d'information entre serveur et client, de manière cachée à l'utilisateur. Ces transferts ralentissent l'affichage et augmentent la charge du réseau; on peut remplacer la plupart des opérations faites *via* le système de bibliothèque RLS par des traitements demandés au serveur avec des procédures.




Le contenu d'une bibliothèque : utiliser la fenêtre Explorer

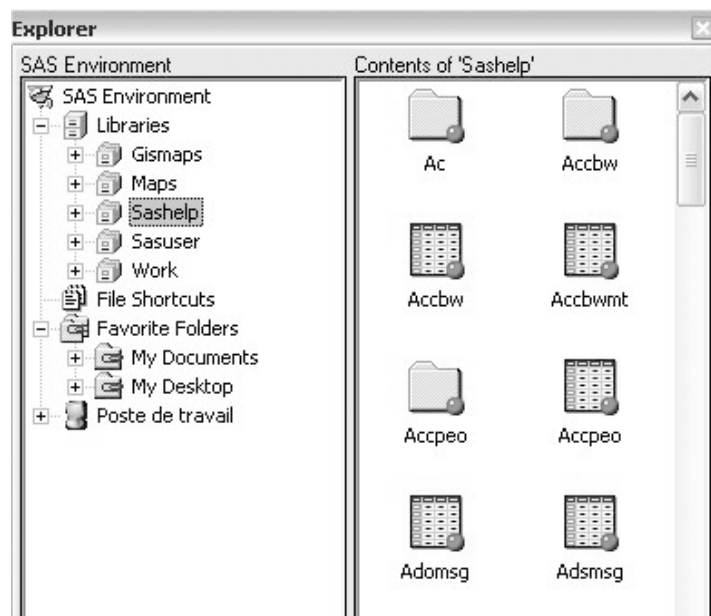
La fenêtre *Explorer* est disponible à partir de la version 8 sous Windows et Unix. Elle permet, plus facilement que sur les précédentes versions (pour lesquelles des



outils analogues existent, en particulier la fenêtre « Lib »), de visualiser les bibliothèques déjà allouées et leur contenu, et d'allouer de nouvelles bibliothèques à travers un assistant.

En cliquant sur l'icône **LIBRARIES**, la liste des bibliothèques allouées s'affiche. Elles sont listées par ordre chronologique d'allocation. Les bibliothèques peuvent avoir des icônes différents en fonction de leur nature :

- une bibliothèque « normale » a un icône de tiroir à classeurs; 
- une bibliothèque « Access To » a un icône agrémenté d'un globe terrestre; 
- une bibliothèque RLS a un icône agrémenté d'un ordinateur et d'un réseau. 

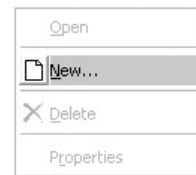


Le menu **VIEW > SHOW TREE** permet d'ajouter à la fenêtre *Explorer* une vision arborescente qui facilite grandement la navigation – mais prend de la place à l'écran. Cette possibilité est illustrée dans la capture d'écran précédente.

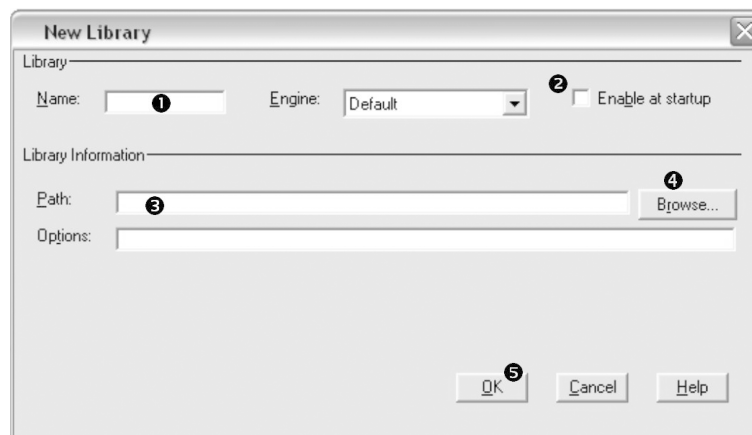
C'est à partir de cet écran que s'ouvre l'assistant d'allocation de bibliothèques.

Assistant de création de bibliothèques

Pour accéder à cet assistant, faire un clic droit dans l'écran précédent, et choisir **NEW** dans le menu contextuel qui s'affiche alors.



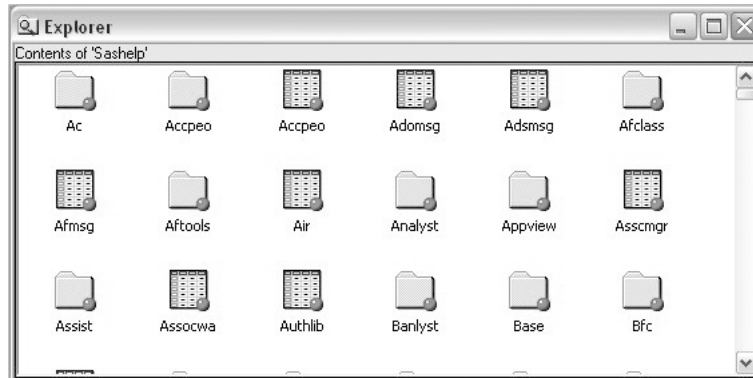
Dans la version 6, il est possible de trouver un assistant similaire en ouvrant l'écran de gestion des bibliothèques puis en cliquant sur le bouton **NEW**. La fenêtre qui s'ouvre alors est identique dans son principe à celle présentée ci-dessous, capturée sur une version 9.1 sous Windows.



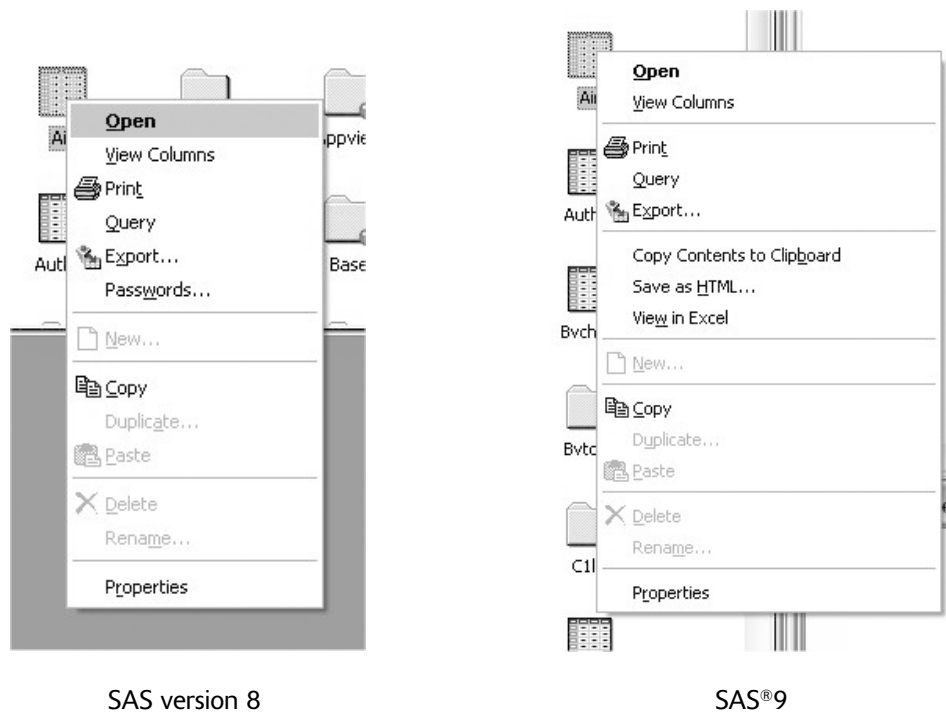
Le nom de la bibliothèque s'indique dans la zone ❶. Si l'on coche la case ❷, la bibliothèque sera allouée à chaque nouvelle session SAS sans autre manipulation de l'utilisateur. Le chemin correspondant à la bibliothèque est soit tapé directement dans le champ ❸, soit choisi à travers la fenêtre de dialogue qui s'ouvre quand on clique sur le bouton ❹. L'allocation de la bibliothèque est entérinée avec le bouton ❺.

Visualiser le contenu d'une bibliothèque

À partir de l'écran listant l'ensemble des bibliothèques allouées, on en « visite » une en cliquant deux fois sur son icône. Ici, le contenu de la bibliothèque SASHELP.



Les différents types de fichiers (décrits plus loin) sont repérés par des icônes différents. Un clic droit sur l'un de ces fichiers permet d'accéder à un menu contextuel qui autorise la navigation dans le fichier, sa copie, sa suppression, etc.



Le choix **OPEN** correspond à l'utilitaire **VIEWTABLE** dont les manipulations seront précisées au paragraphe 2.3; **VIEW COLUMNS** fait également l'objet d'une description dans le même chapitre.

Les autres choix sont laissés à l'appréciation du lecteur. Attention cependant au choix **PRINT** : il envoie une impression du contenu du fichier vers l'imprimante.

mante par défaut, ce qui peut s'avérer peu judicieux dans le cas d'un fichier de grosse taille...

Les bibliothèques prédéfinies de SAS

SAS propose de lui-même, au démarrage d'une session, au moins trois bibliothèques prédéfinies. Ce sont les bibliothèques WORK, SASHELP et SASUSER. L'ensemble des bibliothèques définies dans une session est visible dans la fenêtre Explorer (versions 8 et postérieures), dans la partie **LIBRARIES**.

WORK

WORK est une bibliothèque *temporaire*. Son contenu est **purgé** à la fin de la session SAS. En revanche, cet espace est strictement personnel à l'utilisateur, et deux sessions SAS, même si elles sont démarrées en même temps sur la même machine par le même utilisateur, ne partageront jamais leur WORK : l'espace disque sous-jacent sera toujours différent.

Il n'est pas conseillé de créer dans WORK autre chose que des résultats intermédiaires. Toute donnée stockée dans WORK sera irrémédiablement perdue à la fermeture de SAS, sans qu'un message vienne demander confirmation de la suppression du travail stocké dans cette bibliothèque.

SASHELP

SASHELP est une bibliothèque, parfois partagée entre plusieurs utilisateurs, dans laquelle l'écriture est interdite. Son rôle est d'héberger des données fournies par SAS à des fins d'exercices (comme CLASS, AIR, SHOES, etc.), ainsi que des informations nécessaires au bon fonctionnement du logiciel. On y trouve également des données dites « de référence » qui contiennent la liste des objets de certains types (par exemple, la liste des bibliothèques définies se trouve dans la vue VSLIB).

SASUSER

SASUSER est une bibliothèque personnelle (chaque utilisateur a sa propre SASUSER) permanente (au contraire de WORK). Son contenu n'est donc pas effacé à la fermeture de SAS.

Cette bibliothèque contient entre autres les informations de personnalisation de la session SAS (barres d'outils par exemple). Il est possible d'y écrire ses données, même si, en général, il est préférable d'allouer une bibliothèque séparée pour le stockage des données.

MAPS

Selon la configuration de SAS, une bibliothèque MAPS peut être allouée automatiquement en supplément des trois précédentes. Souvent partagée, interdite

en écriture, cette bibliothèque contient les fonds de cartes fournis par SAS qui représentent la majorité des pays du globe, dont une carte départementale de la France. Ces fonds de cartes ne sont utilisables qu'à travers certains des programmes de cartographie de SAS (module SAS/GRAPH mais pas GIS).



2.1.2 Les tables et les vues

Les données hébergées par SAS prennent le plus souvent la forme de *tables* ou de *vues*. Dans les deux cas, les données sont organisées sous forme tabulaire, avec des lignes et des colonnes. On parlera le plus souvent d'*observations* (pour les lignes) et de *variables* (pour les colonnes). Ce vocabulaire particulier, hérité de la vocation statistique de SAS, souligne aussi que les lignes et les colonnes d'une vue ou d'une table n'ont pas un rôle identique, et ne sont *pas interchangeable* – au contraire d'une feuille Excel par exemple.

Ce vocabulaire originel tend à s'estomper au fil des dernières versions de SAS : les documentations de l'éditeur comme les messages de la fenêtre *Log* panachent désormais les deux dénominations, les utilisant comme synonymes. Ce livre utilise de manière systématique les termes d'observations et de variables pour souligner particulièrement le fait que leurs rôles ne sont absolument pas interchangeables.

Quelle différence entre une table et une vue ?

Si on ouvre une vue et une table SAS, la similitude de l'aspect est déroutante. Pourquoi disposer de deux objets différents si les données sont organisées de la même façon ? C'est en fait le mode de stockage qui diverge entre vue et table.

- Une *table* est une « *photo* » des données : elle stocke des données qui ne changeront plus jusqu'à la création d'une nouvelle version de la table. 
- Une *vue* est un *lien* vers un ensemble de données : elle ne contient qu'une requête, qui sera réexécutée à chaque accès (par un programme ou par l'utilisateur voulant consulter *de visu* son contenu) à la vue. Cette requête permet de récupérer l'information voulue, à partir d'une ou de plusieurs sources de données (tables ou autres). 

L'avantage d'une vue sur une table est double : place réduite pour le stockage (il n'y a que la définition de la requête à conserver), et données perpétuellement à jour sans nécessiter d'intervention de l'utilisateur. La table, en revanche, permet un accès plus rapide aux données, et également de figer dans le temps un état : cela s'avère pratique pour conserver un historique à partir de données mouvantes (données de production par exemple).

Tableau 2.1 – Avantages comparés d'une vue et d'une table

Temps d'accès (utilisation par un programme, visualisation)	Place prise pour le stockage	Rafraîchissement des données
Avantage à la table La vue nécessite <i>systématiquement</i> le temps de réexécuter la requête.	Avantage à la vue La vue ne demande que l'espace pour stocker une requête, soit quelques kilo-octets.	Avantage à la vue La vue utilise toujours la dernière version des données, puisqu'elle va <i>toujours</i> les rechercher.

Il n'est donc préférable de faire appel à une vue que lorsque les données sur lesquelles on travaille changent plus fréquemment que ne sont exécutés les programmes qui les utilisent.

Ainsi, un programme SAS exécuté au quotidien sur des données mensuelles demandera plutôt une table. Au contraire, un programme exécuté à chaque trimestre sur des données rafraîchies à la fin de chaque semaine s'accommodera d'une vue, qui garantit la fraîcheur des données utilisées : ce sont les plus récentes disponibles.

Les observations

On désigne par ce nom les lignes d'une table ou d'une vue SAS. Chaque observation correspond à une unité de comptage; en statistique, on parle d'« individu statistique ». Selon les contextes, chaque observation peut correspondre à une vente, une visite chez un médecin, un contrat, une opération bancaire, un client, un département français, etc.

Chaque observation porte un numéro. Il est séquentiel, et recalculé à chaque nouvelle création de la table.

Les variables

Les variables sont les colonnes d'une table ou d'une vue SAS. Chaque variable correspond à une information connue sur les observations. Cette information reste cohérente pour toutes les observations de la table; par exemple, il n'est pas conforme à l'esprit d'une table SAS (ou d'une vue) que, dans une même variable, on retrouve à certaines observations un code postal, et à d'autres un numéro de téléphone.

Chaque variable est repérée par un faisceau de caractéristiques; certaines sont facultatives, d'autres obligatoires; certaines sont uniques, d'autres non. Chaque variable peut posséder un type, un nom, un label, un format, une longueur, un informat.

Le type

Le type d'une variable est le plus simple à décrire : dans SAS, au contraire d'autres langages, les variables ne sont que de deux types : *numérique* ou *caractère*.

- Le type est numérique quand des calculs (comme une moyenne ou une somme) ont un sens sur cette variable. Par exemple, un montant est numérique; en revanche, un code postal (bien que composé exclusivement de chiffres) n'a pas intérêt à être une variable numérique.
- Une date est souvent définie comme numérique : SAS stocke les dates comme un nombre de jours depuis le 1^{er} janvier 1960.
- Une variable est de type caractère quand l'une et l'autre de ces conditions (date ou calculs) ne sont pas remplies.

Une astuce simple permet de déterminer facilement le type d'une variable quand on consulte une table SAS : si la valeur de la variable est justifiée à gauche, il s'agit d'une variable de type caractère. Si la valeur est justifiée à droite, il s'agit d'une variable de type numérique.

Le nom

Le nom d'une variable doit être choisi de manière unique au sein d'une table ou d'une vue. Ce nom sera l'élément utilisé dans les programmes pour repérer la variable. Il doit donc être relativement parlant, compte tenu des contraintes :

- ce nom ne doit pas dépasser 32 caractères (8 en version 6 de SAS);
- il ne doit comprendre qu'un jeu limité de caractères (lettres non accentuées, chiffres, « blanc souligné » ou *underscore* : « _ »);
- il ne doit pas commencer par un chiffre.

Dans le nom d'une variable, les majuscules et minuscules sont considérées de manière complètement indifférente. L'utilisation d'une alternance entre minuscules et majuscules pour marquer des coupures dans le nom (par exemple, en nommant la variable `NomClient`) n'est que très rarement exploitée par SAS (uniquement dans la procédure PRINT).

Le label

Le label d'une variable est un texte libre de 256 caractères maximum (40 dans la version 6) qui permet de décrire le contenu de la variable, de donner sa définition, de décrire éventuellement ses valeurs, de manière plus complète et lisible que ne peut le faire le nom. Le label intervient pour l'affichage de la table. Il doit donc être porteur d'information pour soigner les restitutions.

Le format

Le *format* est un masque d'affichage des données. Il permet d'assurer une transition entre les valeurs telles qu'elles sont stockées (de manière parfois peu lisible : les dates sont stockées comme un nombre de jours depuis le 01/01/1960) et les valeurs telles qu'elles sont affichées. Le format de SAS fonctionne à l'identique d'un format de cellule dans Excel.

Dans le cas d'une date, le format est presque indispensable pour faire le lien entre la valeur stockée (15 000 par exemple) et la valeur lue (25/01/2001 dans le cas de 15 000).

Tableau 2.2 – Formats les plus courants fournis SAS

Nom générique du format	Description	Exemples		
		Valeur stockée	Format	Valeur affichée
\$x.	Affiche les x premiers signes de la valeur d'une variable de type caractère	ABcdeF	\$3.	ABc
		ABcdeF	\$6.	ABcdeF
		ABcdeF	\$8.	ABcdeF puis deux blancs
\$UPCASEx.	Affiche les x premiers signes <i>en majuscules</i> de la valeur d'une variable de type caractère	ABcdeF	\$UPCASE5.	ABCDE
x.	Affiche un entier sur x chiffres à partir de la valeur d'une variable de type numérique : les formats 2. et 3. sont ici trop courts pour un affichage correct, d'où les réactions de SAS (respectivement des *, et un passage à la notation scientifique)	1234	4.	1234
		1234	3.	1e3
		1234	2.	** et un message dans la fenêtre Log
		1234	6.	1234 précédé de deux blancs
		1234.56	4.	1235
Zx.	Affiche un nombre sur x caractères, en complétant par des zéros à gauche	1234	Z7.	0001234

Nom générique du format	Description	Exemples		
		Valeur stockée	Format	Valeur affichée
x.d	Affiche un nombre sur x caractères, dont d décimales et un caractère séparateur décimal (le point)	1234	7.2	1234.00
		1234.56	7.1	1234.6 précédé d'un blanc
		1234.56	7.2	1234.56
NUMXx.d	Affiche un nombre sur x caractères, dont d décimales et un caractère séparateur décimal (la virgule)	1234	7.2	1234,00
		1234.56	7.1	1234,6 précédé d'un blanc
PERCENTx.d	Affiche la valeur d'une variable comme un pourcentage, en utilisant x caractères pour l'affichage (dont un pour un séparateur décimal (le point), un pour un espace après le nombre, et un pour le signe %)	0.1234	PERCENT7.2	12.34 %
FRACTx.	Affiche un nombre sous forme de fraction d'entiers	0.10	FRACT10.	1/10
DDMMYY10.	Affiche la valeur d'une variable date sous diverses formes	15000	DDMMYY10.	25/01/2001
MMYY57.		15000	MMYY57.	01/2001
YEAR4.		15000	YEAR4.	2001
FRADFWDX.		15000	FRADFWDX.	25 janvier 2001
FRADFWKX.		15000	FRADFWKX.	jeudi 25 janvier 2001
FRADFMN.		15000	FRADFMN.	janvier

En complément de ces formats « natifs », SAS propose, avec un programme (la procédure FORMAT qui sera étudiée au chapitre 5), de définir des formats

personnalisés correspondant aux différents besoins (tranches de revenus, libellés des départements français, etc.).

La longueur

La *longueur* (*length*) d'une variable est la place prévue pour le stockage de chacune de ses valeurs. SAS prévoit par défaut d'utiliser 8 octets pour chaque valeur. Les longueurs autorisées sont :

- de 1 à 32 767 octets pour une variable de type caractère;
- de 2 à 8 octets pour une variable de type numérique. Il est à noter que sous Windows, Unix et OS/2, les longueurs possibles sont comprises entre 3 et 8 octets uniquement.

Pour une variable de type caractère, un octet = un caractère stocké. Le calcul de la longueur nécessaire à une variable est donc simple : c'est la longueur de la plus grande valeur à stocker.

Pour une variable numérique aux *valeurs entières*, le tableau 2.3 donne des ordres de grandeurs. On y voit que le stockage d'une date, par exemple, ne nécessite en général que 3 ou 4 octets, ce qui permet de gagner en taille de la table SAS par rapport aux attributs par défaut. Un booléen (variable vraie ou fausse, valant souvent 0 ou 1) se contentera (largement !) de la taille minimale : 2 ou 3 octets. Pour une variable numérique avec *des décimales*, il est conseillé de conserver la longueur 8.

Tableau 2.3 – Capacités de stockage d'une variable numérique en fonction de sa longueur et du système d'exploitation

Longueur de la variable (en octets)	Plus long entier stocké	
	Windows, Unix, OS/2, OpenMVS Alpha	CMS, VAX, OS/390
2	–	256
3	8 192	65 536
4	2 097 152	16 777 216
5	536 870 912	4 294 967 776
6	137 438 953 472	1 099 511 627 776
7	35 184 372 088 832	281 474 946 710 656
8	9 007 199 254 740 992	72 057 594 037 927 900

L'informat

Un *informat* est un masque de saisie d'une variable. C'est le symétrique du format; il s'applique entre les données telles qu'elles sont lues lors de la création de la table SAS à partir d'un fichier externe, et fait le lien entre la forme lue et la forme stockée de la donnée.

Ainsi, pour reprendre l'exemple d'une date, on peut avoir une donnée stockée sous la forme 20010125 qui désignerait le 25/01/2001. SAS, lui, doit comprendre que c'est une date, et stocker le nombre 15 000 (car le 25 janvier 2001 est le 15 000^e jour depuis le 1^{er} janvier 1960). L'informat permet d'indiquer comment faire cette transformation, en particulier le fait que, dans la valeur lue, les quatre premiers chiffres sont ceux de l'année, les 2 suivants, ceux du mois, et enfin ceux du jour.

Les informats les plus courants sont répertoriés dans le tableau 3.12, lors de l'explication des instructions d'import. La procédure FORMAT, si besoin est, permet de définir des informats personnalisés.

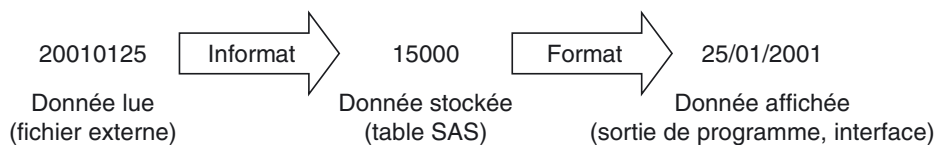


Figure 2.2 – Application d'un format et d'un informat sur une date

2.1.3 Les autres objets créés par SAS

SAS crée également d'autres fichiers à part les tables et les vues.

Les catalogues

Les *catalogues* sont les plus courants. Il s'agit de fichiers d'usage quasi exclusivement interne à SAS, que les autres applications ne peuvent importer (au contraire d'une table SAS, qui peut être lue par d'autres logiciels statistiques par exemple). Un catalogue permet de stocker diverses informations, comme un format défini par l'utilisateur, ou un programme paramétré et compilé à l'avance (macro-programme, voir à ce propos le chapitre 6). Les catalogues hébergent aussi les écrans définis par le module SAS/AF, pour le développement d'applications presse-bouton analogue à Visual Basic.



Les cubes

SAS crée aussi des *bases multidimensionnelles* (MDDB ou *Multi-Dimentionnal DataBase*); ce sont des cubes de données, auxquels on accède par des méthodes de type OLAP. Ces cubes sont disponibles à travers les modules



WebEIS, WebAf ou encore EIS, et sont créés avec la procédure MDDDB incluse dans le module SAS/MDDDB Server. Il s'agit de données pré-agrégées, sur lesquelles les statistiques ont déjà été calculées, et les axes d'exploration déjà définis.

Les fichiers « invisibles » : index et magasins d'items

Si les catalogues et les MDDDB sont visibles dans les bibliothèques (on s'en aperçoit en parcourant la bibliothèque SASHELP par exemple), SAS crée aussi deux types de fichiers que son interface ne montre pas : les index et les magasins d'items.

Les *index* permettent un accès plus rapide à une table (pour une jointure, un filtre) en répertoriant les numéros des observations auxquelles apparaissent les valeurs d'une certaine variable, ou d'un n-uplet de variables. La table correspondante est alors indexée selon cette (ou ces) variable(s). Un index, s'il accélère les traitements (il facilite les jointures et optimise les filtres), est long à créer, et prend de la place sur le disque où la table est stockée.

Les *magasins d'items* (en anglais *Item Store*) sont analogues dans leur fonction à des catalogues : il s'agit de ressources internes que SAS organise en sous-répertoires mais qui ne constituent physiquement qu'un seul fichier. Ils hébergent entre autres des sorties non mises en forme (appelées documents) et des styles pour la mise en forme de ces sorties (appelés *templates* ou modèles).

2.1.4 Les fichiers externes et les filerefs

Même si cela tend à disparaître, il est encore parfois utile de signaler à SAS l'existence d'un fichier dans lequel on a à lire des données ou dans lequel on désire écrire. Ces fichiers sont différenciés de tous ceux que l'on vient de citer – tables, vues, catalogues, etc. par le fait que leur format n'est pas exclusif à SAS. On les regroupe sous l'appellation générique de *fichiers externes*. Il s'agit souvent de fichiers texte, également appelés fichiers plats : leurs extensions sont TXT, DAT, CSV...

Généralement, l'emploi d'un fichier externe dans SAS se fait en citant directement, entre guillemets, son chemin et son nom complet (extension comprise s'il y a lieu). Il peut aussi faire l'objet d'un alias, qu'on appelle un *fileref*, défini par l'instruction FILENAME. *Son nom est régi par les mêmes contraintes que celui d'une bibliothèque*¹.

```
| FILENAME nomFileref "chemin et nom du fichier externe";
```

Quand cette instruction s'exécute correctement, la fenêtre *Log* n'affiche aucun message particulier. Il est d'ailleurs possible de désigner par un FILENAME

1. Cf. section 2.1.1.

un fichier n'existant pas encore. En revanche, le chemin (les répertoires et sous-répertoires) doit déjà exister. Les instructions FILENAME et LIBNAME ne permettent pas, sauf en environnement *mainframe*, de créer des répertoires.

2.2 LES PROGRAMMES SAS

SAS étant principalement un logiciel où il faut programmer, il est temps maintenant de voir quelles sont les principales familles d'instructions qui composent un programme SAS. L'étape Data et les procédures sont les deux familles principales, auxquelles s'ajoutent des instructions satellites regroupées en deux familles : les instructions globales (qui vont avoir un effet durant toute la session SAS) et les instructions ODS (qui permettent l'aiguillage des résultats vers divers formats).

2.2.1 L'étape Data

L'étape Data regroupe un ensemble d'instructions de programmation commençant par une ligne DATA et se terminant par l'instruction RUN ; qui clôt cette étape. Le but principal de l'étape Data est la manipulation *observation par observation* d'un jeu de données. Les données entrantes ou sortantes pouvant être aussi bien une table SAS¹ qu'un fichier externe, l'étape Data offre une gamme d'utilisations assez vaste :

- requête sur une table ou une vue SAS (extraction de certaines observations et de certaines variables) ;
- création de nouvelles variables ;
- importation de fichiers externes ;
- exportation de fichiers externes ;
- fusion de plusieurs tables.

Les principales fonctionnalités de l'étape Data seront détaillées, avec leur syntaxe et des exemples, dans le chapitre 3.

2.2.2 Les procédures

Les *procédures* sont des programmes déjà écrits de SAS, à la syntaxe beaucoup moins libre que celle de l'étape Data, où il faut surtout renseigner un certain

1. Le terme de « table SAS » est à prendre au sens large : tables, vues et contenus de bibliothèques définies avec SAS/ACCESS, vers des bases de données ou des classeurs Excel par exemple (voir paragraphe 2.1.1).

nombre de *paramètres* et faire jouer certaines *options*. Les procédures, qui sont assez nombreuses (aux alentours de 200 sur l'ensemble des modules de SAS), accomplissent chacune une tâche *spécialisée* : tri des données, listing d'une table, création de format, etc.

Chaque procédure commence par une instruction PROC suivie de son nom, et s'achève sur une instruction RUN; qui demande l'exécution de cette procédure. Certaines procédures demandent de surcroît une instruction QUIT; pour clore réellement leur syntaxe.

Le chapitre 3 parle de trois procédures connexes à la manipulation de données avec l'étape Data : la PROC CONTENTS (qui produit la liste des caractéristiques d'une table et le dictionnaire de ses variables), PROC SORT (qui trie une table) et PROC SQL (où l'on programme en langage SQL des fonctionnalités assez analogues à l'étape Data ou à d'autres procédures).

Le chapitre 4 recense et détaille des procédures pour la publication d'informations : la PROC PRINT pour l'édition d'un listing à partir d'une table, la PROC FORMAT pour la création de formats personnalisés, la PROC TABULATE pour les tableaux croisés statistiques, ainsi que quelques procédures graphiques comme GCHART (graphiques circulaires, diagrammes en bâtons), GPLOTT (nuages de points et courbes).

Le chapitre 5 développe pour sa part les procédures de statistiques les plus usuelles : MEANS pour le calcul de statistiques descriptives, FREQ pour l'édition de tableaux de fréquences et les tests du chi-2 et assimilés, CORR (calculs de corrélations) et UNIVARIATE (description de la distribution d'une variable). Ce chapitre comporte également un tableau qui recense les principales procédures nécessaires à une analyse statistique poussée (régressions, ACP, etc.).

Différences entre une procédure et une étape Data

Toutes les procédures rattachées au langage de base¹ ont en commun d'avoir une *vision globale* de la table sur laquelle elles travaillent; cela contraste avec l'étape Data, qui a une vision n'incluant qu'une *observation à la fois*. Par exemple, considérons la table CLIENTS d'une banque :

Pour déterminer, *par client* (i.e. par observation) le montant total des sommes détenues, il faudra faire une *étape Data*. À chaque observation, on cumulera les montants détenus sur les trois types de comptes – compte courant, livret et PEL.

Pour déterminer la somme *totale* déposée par l'ensemble des clients sur leurs Plans Epargne Logement, il faudra utiliser une *procédure* (PROC MEANS par exemple) pour calculer la somme de la colonne la plus à droite de notre table.

1. Sont exclues de ces considérations les procédures IML et SQL, qui conduisent à programmer dans un langage autre que le SAS de base, et où la distinction avec l'étape Data n'est plus vraie.

N° client	Montant compte courant	Montant livret	Montant plan épargne logement
1	8 306,03	143,59	161,18
2	1 47,42	5 109,96	0
3	1 525,39	9 034,18	4 341,00
4	13 162,83	0	17 542,81
5	45,86	1 581,75	15 588,30

Une autre différence majeure entre l'étape Data et une procédure est que la première ne produit pas de sorties dans la fenêtre *Output*, tandis que la seconde, si. Cependant, cette règle souffre de quelques exceptions. Ainsi l'étape Data, à travers l'instruction FILE PRINT (assez rare d'emploi néanmoins) peut envoyer du texte dans la fenêtre *Output*. À l'inverse, certaines procédures (FORMAT, SORT, DELETE pour ne parler que des plus courantes) ne produisent par défaut aucun résultat dans la fenêtre *Output*.

Enfin, si une procédure ne peut, en général, traiter qu'une table à la fois, l'étape Data peut pour sa part lire plusieurs sources de données. Cette dernière est donc un passage obligé pour combiner plusieurs tables en vue du traitement par une procédure.

2.2.3 Les instructions globales

Sous le nom d'instructions globales, on regroupe toutes les instructions qui ont un effet tout au long de la session SAS. Les instructions LIBNAME et FILENAME déjà vues sont des instructions globales. On y compte également les titres et pieds de page, les options système et les *goptions*.

Les titres sur 10 niveaux (instructions TITLE1, TITLE2, etc., jusqu'à TITLE10) permettent de faire précéder les sorties de messages sur leur nature, leur date d'édition, etc.; les pieds de pages (*footnote*) fonctionnent à l'identique des titres : on peut faire suivre un résultat d'un texte de 10 lignes maximum (instructions FOOTNOTE1 etc.).

```
TITLE1 "texte de la 1re ligne de titre";
FOOTNOTE1 "texte de la 1re ligne de pied de page";
... traitement(s) ...
TITLE;
FOOTNOTE;
```

Les deux dernières instructions permettent de réinitialiser (= vider) toutes les lignes de titres et de pieds de page. Sans ces instructions, l'effet des titres et pieds de pages est permanent, c'est-à-dire que toutes les sorties en seront affublées jusqu'à la fin de la session SAS.

Les *options* (ou *options système*) modifient le fonctionnement de SAS – emplacement des catalogues de formats, comportement en cas de format inconnu, compression des tables systématique ou non, largeur et longueur d'une page logique de la fenêtre *Output*, etc.

Tableau 2.4 – Principales options de SAS

Nom de l'option	Effet	Syntaxe (exemple)
COMPRESS	Compression systématique des tables SAS créées : NO (par défaut), ou YES.	OPTION COMPRESS = NO ;
FMTSEARCH	Liste des bibliothèques (allouées au préalable) contenant un catalogue de formats personnalisés	OPTION FMTSEARCH = (WORK FMTPERSO) ;
FMTERR	Par défaut, SAS refuse d'utiliser une table appelant des formats inconnus (c'est l'option FMTERR) ; on désactive ce comportement avec NOFMTERR.	OPTION NOFMTERR ;
		OPTION FMTERR ;
DATE	SAS affiche par défaut la date et l'heure d'ouverture de la session courante en tête de chaque page de sortie (fenêtre <i>Output</i> , ODS RTF) ; on les supprime avec NODATE.	OPTION NODATE ;
		OPTION DATE ;

Les options graphiques (ou *goptions*) sont identiques dans leur principe aux options système du tableau 2.4, mais elles n'influencent que sur l'aspect des graphiques produits par le module SAS/GRAPH. Les principales *goptions* seront vues au chapitre 4.

Toutes ces instructions sont citées en dehors des procédures et des étapes Data.

2.2.4 L'Output Delivery System

L'*Output Delivery System* (ODS en abrégé) est disponible depuis la refonte du mode de fonctionnement des procédures (version 7). SAS est passé d'un fonctionnement où les procédures étaient chargées d'éditer elles-mêmes leurs résul-

tats à une délégation de tout ce qui n'est pas calcul à l'ODS. Cela permet de s'affranchir de nombreuses contraintes, de syntaxe en particulier, tout en facilitant l'envoi des résultats, de SAS vers des formats jusqu'alors inusités par ce logiciel : Word, Excel, PDF, page web (fichier HTML), etc.

La figure 2.3. illustre le mode de fonctionnement d'une procédure dans les versions supportant l'ODS (toutes les versions postérieures à la 6.12).

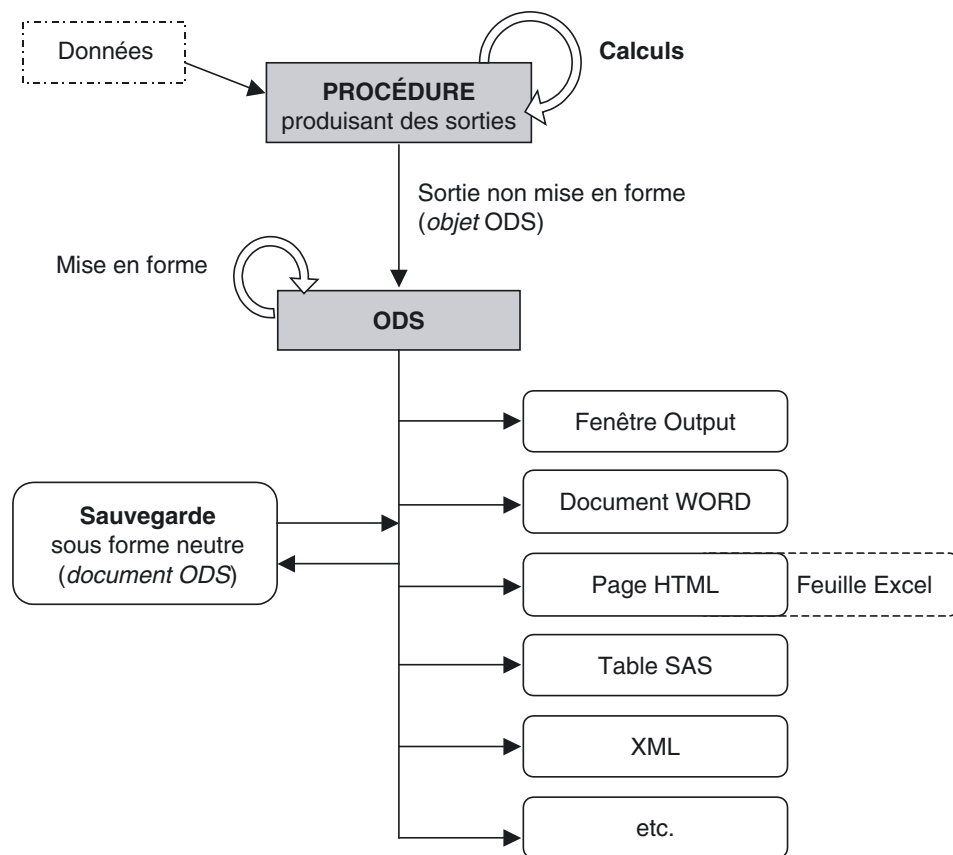


Figure 2.3 – Fonctionnement d'une procédure avec l'ODS

Chaque procédure peut produire un ou plusieurs *objets ODS*. Ceux-ci sont ensuite traités par l'ODS. Les noms des objets ODS dépendent de la procédure, mais pas des données sur lesquelles la procédure s'exécute. Ainsi, l'objet ODS produit par une procédure MEANS (chapitre 5) s'appellera-t-il toujours SUMMARY.

Il est possible de n'envoyer que certains objets à l'ODS, ou à l'une de ses destinations. Pour cela, on utilisera une des syntaxes suivantes :

```

ODS <destination> SELECT nomObjetTraité;
ODS <destination> EXCLUDE nomObjetNonTraité;
  
```

Si on ne précise pas de destination, tous les formats de documents de l'ODS (y compris LISTING, c'est-à-dire le texte dans la fenêtre *Output*) sont concernés.

Syntaxe de l'ODS : les différentes destinations

```
ODS typeFichierProduit < option(s) >;
...
procédure(s) sans option NOPRINT
...
ODS typeFichierProduit CLOSE;
```

Dans la syntaxe des instructions de l'ODS, on distingue deux parties : une instruction *ouvrante* qui demande l'envoi vers un certain type de fichier des résultats; une instruction *fermante* (avec CLOSE) qui termine cet envoi.

Le type de fichier produit – on parle de *destination ODS* – est représenté par un mot clé (les plus courants sont donnés dans le tableau 2.5). Les options peuvent être nombreuses, la principale est généralement FILE derrière laquelle on donne le nom et l'emplacement du fichier produit.

Tableau 2.5 – Principales destinations ODS

Nom de la destination	Type de fichier produit	Disponible à partir de la version...
LISTING	Fenêtre <i>Output</i> (destination activée par défaut)	7.0
HTML	Page Internet, feuille Excel (à partir d'Excel 97)	7.0
RTF	Document Word	8.0
OUTPUT	Table SAS	8.0
PRINTER	Envoi à une imprimante, document Postscript	8.0
PDF	Document PDF	8.1
MARKUP	Document XML	9.0 expérimental en 8.2
DOCUMENT	Sauvegarde de la sortie sous forme neutre	9.0 expérimental en 8.2
GRAPHICS	Active la production de graphiques par les procédures statistiques; doit être utilisé conjointement avec certaines autres destinations ODS (HTML, RTF, DOCUMENT, ...)	9.1

Pour illustrer l'apport de l'ODS, le programme suivant est exécuté avec et sans l'instruction ODS HTML. Le corps du programme utilise une procédure MEANS pour calculer la moyenne et la somme des montants vendus à l'ensemble des clients.

Exemple 2.5 – ODS (1) : création d'une page HTML à partir de SAS

```
ODS HTML FILE = "c:\temp\montants_ventes.htm";
PROC MEANS DATA = base.ventes MEAN SUM MAXDEC = 2;
    VAR mt;
RUN;
ODS HTML CLOSE;
```

```

The SAS System
The MEANS Procedure
Analysis Variable : mt Montant HT des achats par client

```

Mean	Sum
3266.70	49000.57

Sortie standard de SAS
(fenêtre Output)

The SAS System	
The MEANS Procedure	
Analysis Variable : mt Montant HT des achats par client	
Mean	Sum
3266.70	49000.57

Sortie avec ODS HTML
(fichier montants_ventes.htm)

Création de tables SAS via l'ODS

De nombreuses syntaxes spécifiques existent en fonction des destinations choisies. Nous n'en présenterons qu'une seule – la destination Output –, celle permettant la production d'une table SAS en sortie de n'importe quelle procédure produisant des sorties. Pour cela, il est nécessaire de connaître le nom de l'objet à récupérer.

Dans les chapitres suivants, chaque procédure sera présentée avec les noms des principaux objets qu'elle génère. Pour connaître le nom d'un objet non cité dans ces pages, il faut exécuter le programme encadré des instructions ODS TRACE ON; et ODS TRACE OFF; pour faire apparaître dans la fenêtre Log un message.

Exemple 2.6 – ODS (2) : récupération du nom des objets créés par un programme

```
ODS TRACE ON;
PROC MEANS DATA = base.ventes MEAN SUM MAXDEC = 2;
    VAR mt;
RUN;
ODS TRACE OFF;
```

L'exemple 2.6 génère le message suivant dans la fenêtre Log :

```

Output Added :
-----
Name :      Summary
Label :     Summary statistics
Template :  base.summary
Path :     Means.Summary
-----

```

Comme le montre cet extrait de la Log, le nom de l'objet généré par la procédure MEANS est bien SUMMARY.

Une fois ce nom connu, la syntaxe nécessaire à la production d'une table SAS est :

```
| ODS OUTPUT nomObjet = nomTableSAS;
```

Dans notre exemple, pour récupérer les statistiques produites dans une table du nom de WORK.STATS_MONTANT, on écrira le programme suivant.

Exemple 2.7 – ODS (3) : création d'une table SAS à partir d'une procédure

```

ODS OUTPUT summary = work.stats_montant;
PROC MEANS DATA = base.ventes MEAN SUM MAXDEC = 2;
    VAR mt;
RUN;
ODS OUTPUT CLOSE;

```

2.2.5 Les commentaires

Il est essentiel d'ajouter au programme un nombre raisonnable de commentaires pour mieux en décrire le but, le fonctionnement et les éventuelles subtilités.

Dans SAS, les commentaires seront de préférence encadrés entre les signes /* et */. Ces commentaires peuvent être constitués de plusieurs lignes. Le texte des commentaires est entièrement libre : SAS ne cherche pas à l'interpréter.

Dans le cadre du test d'un programme, il peut être pratique d'en commenter des parties pour détecter une erreur plus facilement.

Exemple 2.8 – COMMENTAIRES : exemple de commentaire

```
| /* ceci est un commentaire */
```

2.2.6 Autres choses à savoir

Les programmes SAS ne sont pas influencés par la présence d'espaces surnuméraires ou de retours à la ligne dans la typographie. Et, à l'exception des chaînes de caractères citées entre guillemets, il n'y a pas de distinctions opérées entre les caractères minuscules et majuscules. Tirez-en parti pour produire des programmes lisibles plus facilement :

- aérez votre programme, avec au maximum une instruction par ligne;

- indentez vos programmes pour bien montrer les niveaux d'emboîtement;
- choisissez une convention distinguant la syntaxe SAS des valeurs liées aux données; par exemple, majuscules pour tous les mots-clés de SAS, et minuscules pour le reste;
- commentez votre programme de manière régulière et mesurée. Trop de commentaires noient le code du programme. Trop peu de commentaires le rendent excessivement compliqué à relire, ou à transmettre à un tiers.

2.3 NE CROYEZ PAS CE QUE VOUS VOYEZ À L'ÉCRAN !

Cette partie a pour but de mettre en garde le programmeur contre certains mauvais côtés de la mise en forme de l'information par SAS. Le programme à rédiger dans SAS fait toujours référence aux **noms** des variables (et non à leur label), et les diverses opérations sur les données utilisent les valeurs **stockées** de chaque variable, et non les valeurs affichées à travers un format.

Pourtant, quand on veut consulter le contenu d'une table, en particulier à travers la fenêtre Explorer et l'utilitaire appelé **VIEWTABLE** qu'elle inclut, les tables portent les labels en tête de colonne, et les valeurs sont parées de leur format par défaut.

Voici par exemple une table **CLIENTS** telle que SAS l'affiche...

VIEWTABLE: Base.Clients					
	Identifiant client	Nom du client	Prénom du client	Date de naissance du client	Code postal de résidence du client
1	1	Adebayor	Emmanuel	23/08/76	06100
2	2	Simeoni	Albert	21/05/52	75013
3	3	Berquebled	Chantal	.	.
4	4	Dos Santos	Sylvain	12/12/65	45300
5	5	Mellé	Lebelle	01/04/91	22000

... et telle qu'elle est réellement stockée, ce qui doit intéresser davantage le programmeur.

VIEWTABLE: Base.Clients					
	numClient	nom	prenom	dtnais	cpostal
1	1	Adebayor	Emmanuel		6079 06100
2	2	Simeoni	Albert		-2781 75013
3	3	Berquebled	Chantal		.
4	4	Dos Santos	Sylvain		2172 45300
5	5	Mellé	Lebelle		7361 22000

Noter en particulier :

- que les en-têtes de colonnes ont changé;
- que la date de naissance stockée est différente de celle affichée.

Label et nom de variable

La bascule entre nom de variable et label dans **VIEWTABLE** s'avère très simple. Il suffit d'aller dans le menu **VIEW** puis de choisir **COLUMN NAMES** au lieu de **COLUMN LABELS** qui est coché par défaut.

Column Name	Type	Length	Format	Informat	Label
numClient	Number	8			Identifiant client
nom	Text	20			Nom du client
prenom	Text	15			Prénom du client
dtnais	Number	8	DDMMYY8.		Date de naissance du client
cpostal	Text	5			Code postal de résidence du client
sexe	Text	1			Sexe du client F/H
sitfam	Text	3			Situation familiale du client
nbenf	Number	8			Nombre d'enfants du client
email	Text	1			Le client peut être contacté par email O/N
courrier	Text	1			Le client peut être contacté par courrier O/N
telephone	Text	1			Le client peut être contacté par téléphone O/N

SAS version 8

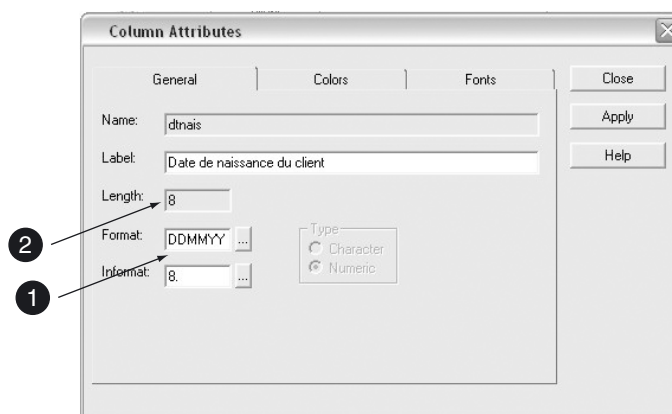
Column Name	Type	Length	Format	Informat	Label
numClient	Number	8			Identifiant client
nom	Text	20			Nom du client
prenom	Text	15			Prénom du client
dtnais	Number	8	DDMMYY8.		Date de naissance du client
cpostal	Text	5			Code postal de résidence du client
sexe	Text	1			Sexe du client F/H
sitfam	Text	3			Situation familiale du client
nbenf	Number	8			Nombre d'enfants du client
email	Text	1			Le client peut être contacté par email O/N
courrier	Text	1			Le client peut être contacté par courrier O/N
telephone	Text	1			Le client peut être contacté par téléphone O/N

SAS version 9

Une autre solution consiste, si l'on connaît déjà bien les valeurs des données et que seuls les noms des variables nous intéressent, à ouvrir une fenêtre qui donne toutes les variables avec leurs attributs. On l'obtient en faisant un clic droit sur la table SAS dans la fenêtre *Explorer* puis en cliquant sur VIEW COLUMNS. Cette fenêtre recense les informations sur les variables de la table, avec entre autres leur nom et leur label.

Valeur et format

Arriver à visualiser la valeur derrière un format est légèrement plus complexe. Sans programmation, et sans sortir de la fenêtre **VIEWTABLE**, il faut cliquer deux fois sur l'en-tête de la variable concernée. Une fenêtre s'ouvre alors, qui contient également les caractéristiques de la variable, mais avec cette fois la possibilité de les modifier (ce qui n'est pas le cas du **VIEW COLUMNS** précédent).



Dans cet écran, le champ **FORMAT** peut voir sa valeur modifiée. On doit y écrire, en lieu et place du format (❶) utilisé par défaut par cette variable, l'un des deux formats suivants :

- BEST12. s'il s'agit d'une variable numérique;
- \$ suivi de sa longueur (champ **LENGTH**, ❷) suivi d'un point, par exemple \$7., s'il s'agit d'une variable de type caractère.

Cliquer ensuite sur **CLOSE**. Les changements de formats sont appliqués immédiatement, mais sont perdus lors de la fermeture de la fenêtre Viewtable : les formats associés par défaut sont à nouveau appliqués.

2.4 LE SCHÉMA DE TRAVAIL AVEC SAS

Par rapport à d'autres langages de programmation, SAS a l'inconvénient de proposer systématiquement les sorties qu'il produit, et de ne pas arrêter l'exécution d'un programme à la première erreur rencontrée. Comme ses programmes sont découpés en étapes (étape Data et procédures), chacune est considérée comme autonome, et une erreur dans une étape n'entraînera que la non-exécution de cette étape. Les étapes suivantes sont quand même exécutées, ce qui peut entraîner des cascades d'erreurs, ou des résultats erronés. Il convient donc d'être extrêmement vigilant lorsqu'on programme dans SAS. En particulier, la *lecture de la fenêtre Log* est une phase *fondamentale*, qui doit toujours précéder la consultation des diverses sorties produites – fenêtre *Output*, table SAS, fichiers externes.

Les étapes d'un travail avec SAS sont donc les suivantes :

1. *mise au clair du problème*; recensement des données nécessaires et des données disponibles;
2. *allocation des bibliothèques* nécessaires pour utiliser les données voulues;
3. *rédaction du programme*, constitué d'une alternance d'étapes Data et de procédures selon le résultat souhaité. Il est conseillé de *tester* (étapes 4 à 8) *au fur et à mesure* le programme, en le compliquant progressivement, de façon à le déboguer plus facilement;
4. *purge de la fenêtre Log* des messages qu'elle contient;
5. *exécution du programme*; SAS exécute le programme de manière linéaire, en traitant chaque étape (procédure ou étape Data) à son tour;
6. *consultation de la fenêtre Log*. Attention, SAS affiche d'emblée les derniers messages. Remonter *en haut* de la fenêtre *Log* pour prendre les éventuels problèmes les uns après les autres dans leur ordre d'apparition;
7. *correction* d'éventuels problèmes et retourner en 4;
8. s'il n'y a pas de problèmes, *consultation des sorties* (tables, fenêtre *Output*, autres fichiers créés par l'ODS ou un export).

Ce qu'il faut retenir

Pour travailler avec SAS, il est nécessaire de disposer de données. Celles-ci seront vues à travers un système de *bibliothèques* (contenant logique pour une ou plusieurs sources de données, pas forcément créées par SAS). Une bibliothèque est généralement déclarée par l'instruction LIBNAME.

À l'intérieur d'une bibliothèque, les objets les plus courants sont des tables, composées d'*observations* (les lignes) et de *variables* (les colonnes). Chaque variable possède au minimum un *nom* et un *type*, numérique (y compris les dates) ou caractère. On peut également lui adjoindre un *label* et un *format*, qui servent à soigner l'affichage et le rendu de l'information.

Les programmes SAS se composent de diverses instructions, organisées en étapes : on distingue l'étape *Data* et les *procédures*. La première opère ligne à ligne sur une ou plusieurs tables ou fichiers lus; les secondes ont une vue d'ensemble de la table et sont nécessaires pour des tâches telles que des opérations statistiques (calculs de moyennes aussi bien que régression), ou plus prosaïquement un tri des données.

Autour de ces étapes gravitent diverses instructions libres, permettant de jouer sur les titres des résultats, le comportement de SAS ou encore la mise en forme des sorties à travers le principe de l'*ODS*. Ce dernier permet de générer à la demande des sorties de diverses formes, dont des documents Word, des pages Internet (qui peuvent être lues par Excel pour les versions les plus récentes de ce logiciel) ou des tables SAS.

Enfin, la manipulation des données de SAS par le programmeur doit se faire avec une extrême prudence, puisque labels et formats déguisent à l'affichage ce qui est la réalité des données. Or le programme demande, lui, qu'on évoque les noms des variables (et non les labels) et les valeurs stockées (et non celles affichées à travers le format).

Le dernier conseil à un programmeur SAS est de ne pas se laisser attirer par les sorties, que SAS propose spontanément au regard lorsqu'il en produit. Mais bien d'aller systématiquement consulter la fenêtre *Log* et de s'assurer qu'aucune des étapes de son programme n'a connu de problème majeur ou inattendu.