APPLICATIONS MÉTIERS

REPORTING AVEC SAS

Mettre en forme et diffuser vos résultats avec SAS 9 et SAS 9 BI

SUPPLÉMENTS en ligne

Olivier Decourt

Préface de Philippe Letren

DUNOD

REPORTING AVEC SAS

Mettre en forme et diffuser vos résultats avec SAS 9 et SAS 9 BI

REPORTING AVEC SAS

Mettre en forme et diffuser vos résultats avec SAS 9 et SAS BI

Olivier Decourt

Consultant et formateur sur SAS

Préface de

Philippe Letren

Responsable du Centre de compétences SAS de la Banque de France

DUNOD

Illustration de couverture :
Panoramique — Champ de lavande provençal © Marc LOBJOY-Fotolia.com.

ISBN 978-2-10-053575-0

Préface

SAS est, à l'origine, un langage de programmation créé en 1976 par Jim Goodnight, californien ayant suivi des études de mathématiques et de statistiques et qui était « tombé amoureux de l'informatique ». SAS est resté longtemps le leader des logiciels statistiques mais, en bon californien, Jimmy Goodnight n'a pas raté la nouvelle déferlante de l'informatique, il surfe aujourd'hui sur celle de la « Business Intelligence ».

La traduction littérale de la Business Intelligence pourrait être « renseignement des affaires » ; on parle en français d'« Informatique Décisionnelle ». Cette subtilité linguistique est caractéristique de l'approche française basée sur l'objectif, c'est-à-dire l'aide à la décision, alors que les Anglo-Saxons eux sont attachés à l'approche intellectuelle de la démarche : la recherche d'informations pertinentes dans la masse énorme des données gérées dans les entreprises.

SAS, acteur majeur de la révolution BI, est désormais un logiciel qui parle à tous, et que tous peuvent questionner. Il est donc hors de question qu'il livre encore des réponses vagues, inesthétiques ou mal présentées.

C'est le sujet même de ce livre : trouver les bonnes informations, découvrir les bonnes corrélations, comprendre les comportements passés, prévoir ceux à venir est vain si l'on ne sait pas ou si l'on ne peut pas présenter de manière claire et pertinente ces résultats. « Ce qui se conçoit bien s'énonce clairement », disait Boileau. La réciproque de Boileau est encore plus vraie ici : on ne comprend bien que ce qui est bien énoncé, bien présenté.

Olivier Decourt nous propose ici un livre didactique, riche et ambitieux, dans lequel les lecteurs trouveront beaucoup de réponses ou de perspectives. Le livre com-

porte notamment de nombreux exemples de code SAS qui viennent illustrer de manière réutilisable les propos tenus.

Ce livre apporte non seulement des réponses techniques ou fonctionnelles adaptées mais aussi des explications claires et accessibles. Olivier Decourt sait et aime transmettre. Son expérience de formateur, tant dans les milieux professionnels qu'universitaires, se sent dès la lecture des premiers paragraphes.

Il était utile de commencer ce livre sur l'ODS – que l'on retrouve en filigrane dans tous les chapitres suivants, car il sous-tend la production de tous les résultats à travers SAS. L'ODS, dont les initiales signifient « *Output Delivery System* » trouve ici une nouvelle traduction explicite : « Organiser et diffuser des sorties ».

Cet ouvrage est destiné à tous ceux qui, dans les entreprises utilisant SAS 9 ou envisageant de le mettre en place, sont intéressés par une diffusion intégrée des connaissances : le même logiciel récupère les données, les traite et les diffuse. Naturellement les lecteurs les plus concernés sont ceux qui sont ou seront amenés à écrire des programmes SAS : étudiants et professionnels, statisticiens et informaticiens.

La clarté et la qualité pédagogique de l'écriture permettent au lecteur de mieux s'approprier l'outil SAS, de solliciter sa créativité pour améliorer l'efficience globale du système.

Philippe LETREN
Responsable du Centre de compétences SAS de la Banque de France,
chargé de la mise en place de SAS 9 BI.

Table des matières

Avant-propos

Pourquoi ce livre ?

Le thème de ce livre peut se résumer en une phrase : *Comment faire des sorties SAS dont on ne penserait pas qu'elles viennent de SAS ?* Chez nombre de mes clients, j'entends les mêmes phrases : « je me sers de SAS pour récupérer mes données et les agréger. Mais pour la mise en forme, les tableaux et les graphiques, je passe sous Excel ».

Est-ce optimal ? Pas forcément. Ça l'est pour un besoin ponctuel, urgent, mais moins dès que l'on souhaite automatiser. Car alors, on devra jongler entre du langage macro côté SAS, et du Visual Basic côté Excel.

Il me paraît bon de connaître l'étendue réelle des capacités de mise en forme de SAS. Depuis la version 8, elles sont *énormes*. Peu de logiciels peuvent revendiquer la même capacité à prendre des données brutes, à leur faire subir divers outrages statistiques, pour enfin les diffuser sous les formes les plus diverses. En fait, il me semblerait encore meilleur de connaître ces capacités *et de s'en servir*. Le but de ce livre est de vous y amener.

Qu'apprend-on dans ce livre ?

Nous parlerons ici de *reporting*, c'est-à-dire de mise à disposition d'information. Concrètement, le reporting se compose d'éléments de trois catégories : des listes, des tableaux et des graphiques. Avec une approche pratique (les pages qui suivent comportent plus d'une centaine d'exemples), nous verrons comment produire tous types de sorties *uniquement* avec le logiciel SAS.

Cet ouvrage présente SAS dans son aspect *Business Intelligence* ou BI (encore appelée « informatique décisionnelle »). Tout d'abord dans l'aspect « diffusion de

l'information », qui est le socle de la Business Intelligence. Tout le travail présenté dans les chapitres à venir tend à rendre intelligibles les données contenues dans des tables SAS. Mais ce livre traite aussi très concrètement du fonctionnement de la plate-forme appelée « SAS 9 BI », en particulier dans la production d'applications stockées (*stored process*), dans le chapitre 7.

La version 9.1.3 a été utilisée pour mettre au point tous les programmes. Toutes les syntaxes « expérimentales » (dont la forme est susceptible de changer sans préavis ni support d'une version à l'autre) sont regroupées dans le chapitre 8. On trouve ici une palette de fonctionnalités, des plus basiques aux plus expertes. Tout ce livre constitue ainsi une boîte à outils dans laquelle on pourra piocher.

La documentation existante sur ces sujets est nombreuse, mais majoritairement anglophone, et éparpillée. Les informations sur l'ODS ne sont pas toutes reprises dans la documentation en ligne de SAS, il y en a une partie sur le site de l'éditeur, et une autre partie encore éclatée dans les contributions aux divers clubs d'utilisateurs à travers le monde.

À qui profite ce livre ?

Il ne s'agit donc pas d'un livre s'adressant à un public forcément très spécialisé. Nous nous adressons ici néanmoins à des personnes qui programment, d'un niveau intermédiaire à avancé. Des débutants pourront trouver certains passages assez complexes, mais l'évolution au sein de chaque chapitre se veut progressive.

Cet ouvrage n'a pas vocation à remplacer l'aide en ligne du logiciel : il n'est absolument pas exhaustif, et aurait pu être aisément deux à trois fois plus long ! La priorité a été donnée aux situations réelles, aux options les plus couramment utiles, à l'expérience vécue. La présence d'une option « exotique » qui ne solutionne qu'un problème très rare n'est absolument pas garantie. Mais son absence de ces pages ne signifie pas qu'une telle option n'existe pas !

Quelle est l'organisation de ce livre ?

Le contenu du livre se divise en cinq grandes parties.

- *La mise en forme* de tous types de sorties est dans les chapitres 1 (*ODS*) et 2 (*Mise en forme avancée*). Dans SAS, une série de commandes autour de l'ODS (*Output Delivery System*) gère la diffusion des sorties ; elle est décrite dans le chapitre 1. Le chapitre 2 indique comment incruster à la demande des caractères spéciaux et des « fonctions » telles que le numéro de page.
- *Les procédures* sont dans les chapitres 3 (*Reporting au format texte*) et 4 (*Graphiques*). Trois procédures gèrent avec souplesse cet aspect des traitements : Print, Tabulate et Report. Chacune est décortiquée, en particulier dans ses interactions avec l'ODS, dans le chapitre 3. Les graphiques sous SAS ont mauvaise presse : ils sont souvent présentés comme laids et inertes. Cependant, un langage graphique extrêmement riche (décrit dans le chapitre 4), permet de produire courbes, camemberts, bâtons, bulles, en définissant très précisément les légendes et les axes.

- Les *techniques plus avancées* de mise en forme sont dans les chapitres 5 (*Style*) et 6 (*Réorganisation des sorties*). On peut personnaliser de manière pérenne une charte graphique utilisée par SAS, comme le montre le chapitre 5. On peut également réorganiser l'ordre des sorties et y ajouter des notes *a posteriori* de leur production (chapitre 6).
- *L'intégration de ces programmes dans l'univers SAS BI* est décrite au chapitre 7. Cette partie montre comment on utilise le contenu des six premiers chapitres selon l'interface que l'on a avec SAS : le « client lourd » traditionnel, *SAS Enterprise Guide*, et comment on utilise l'ODS et le langage graphique dans des applications stockées (le moyen technique de mettre à disposition de tous, *via* Excel ou Internet, des codes SAS pré écrits).
- *Les nouveautés à venir* sont dans le chapitre 8. Produire des tableaux et des textes avec une mise en forme très ajustable, des graphiques esthétiquement irréprochables ? Deux fonctionnalités expérimentales de la version 9.1.3 qui illustrent les futures orientations de SAS en matière de *reporting* sont présentées dans ce dernier chapitre, ainsi qu'une nouvelle procédure graphique pour les tableaux de bord introduite dans la version 9.2.

Remerciements

Je voudrais enfin remercier du fond du cœur tous ceux sans qui ce livre n'aurait pas son aspect actuel, voire pas existé du tout :

- Vanessa, pour son soutien, son encouragement et sa patience.
- Jean-Luc BLANC et Carole TROCHU aux éditions Dunod pour leur confiance et leur professionnalisme.
- Mes très experts collègues formateurs Hugues GÉRARD, Cyrille DUFILS et Lydia STURMA d'Educasoft Formations, qui ont participé à ce projet en relisant patiemment et scrupuleusement les épreuves.
- Mon maître ès-SAS Michel PALLE (Insee, DR de Bretagne) pour sa relecture, son enthousiasme communicatif et son humour ravageur.
- Le méticuleux Bernard GESTIN (rectorat de l'académie de Rennes), grand connaisseur de SAS au regard acéré (merci pour les relectures impitoyables).
- Mes collègues du centre de compétence SAS à la Banque de France, ainsi que l'ensemble des « sasseurs » de la Banque de France avec qui j'ai pu échanger sur ces sujets.
- La très altruiste Cynthia ZENDER (SAS Institute USA) dont l'acharnement à faire la promotion de la procédure Report a fini par emporter mon adhésion.

Et un remerciement immense et tout particulier à Grégory BECQUEMBOIS (Epsilon Consulting), formateur et consultant expert de SAS BI. Le chapitre 7 lui doit énormément. Ma connaissance de plusieurs pans de SAS encore davantage.

Olivier DECOURT
Formateur et consultant indépendant sur SAS depuis 2002

1

ODS : organiser et diffuser des sorties

Objectifs

Depuis l'éphémère version 7 de SAS, l'ODS (*Output Delivery System*) permet de produire des sorties de tous formats : tables SAS, pages HTML, documents RTF/Word, documents PDF et PostScript, fichiers XML, etc. Le but de ce chapitre est de vous permettre de comprendre l'ODS et de produire des sorties mises en forme. Le fonctionnement de l'ODS s'articule autour de morceaux élémentaires de sorties appelés *objets*. Chaque objet porte un nom qui aide à le manipuler.

Le démarrage d'un *document* (un fichier) généré par l'ODS se fait à travers une instruction ouvrante ; une instruction de fermeture la complète plus tard dans le programme, une fois qu'une ou plusieurs procédures ont généré le contenu du document. Une *destination* ODS est un type de fichier que l'ODS sait produire – table SAS, document PDF, fichier HTML, etc. Pour chaque destination, certaines précautions particulières sont à prendre ; des options spécifiques sont également associées à certaines destinations. Elles vous seront détaillées dans la deuxième partie.

Enfin, des options système de SAS, ainsi que des instructions réservées à l'ODS, influent sur l'esthétique du résultat obtenu. Ces options seront détaillées dans la dernière partie de ce chapitre.

1.1 DES UNITÉS ÉLÉMENTAIRES DE SORTIES : LES OBJETS ODS

Les *objets* ODS sont la base de la communication entre les procédures et l'ODS. Il s'agit de résultats qui ont été créés par une procédure mais dont la mise en forme n'est pas encore figée. Plusieurs éléments de mise en forme (les modèles, voir chapitre 5) seront associés à cet objet par l'ODS afin de pouvoir l'afficher et le diffuser dans les formats requis par l'utilisateur.

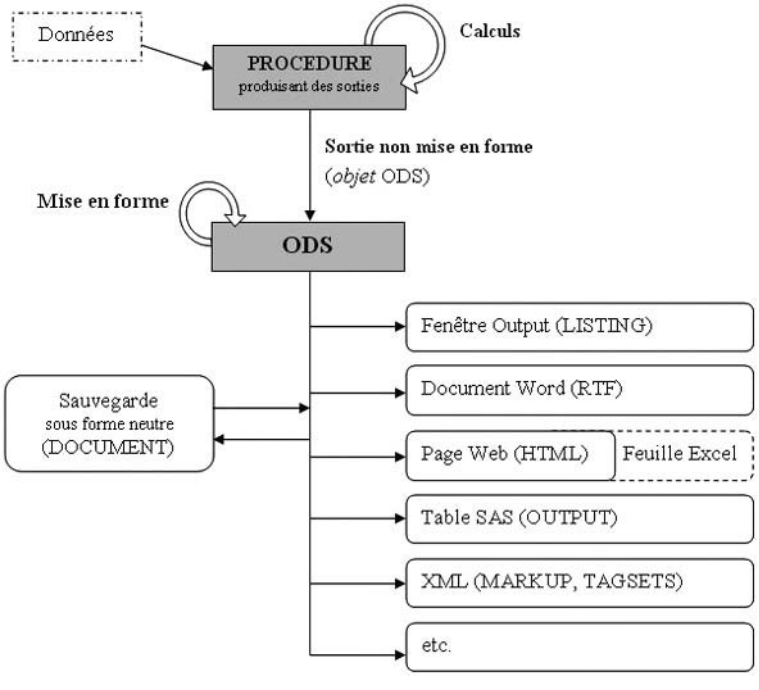


Figure 1.1 — Fonctionnement global de l'ODS

Un objet est spécifique à la procédure qui le crée ; il porte un nom qui le caractérise et permet de le manipuler. Ce nom n'est lié qu'à la procédure qui génère l'objet et non pas aux données qui sont lues par la procédure.

Récupérer le nom d'un objet requiert des manipulations spécifiques ; c'est une information nécessaire pour deux usages : la diffusion sélective des sorties d'une procédure, et la création d'une table SAS à partir d'une procédure.

Attention : si la procédure SAS contient l'option `NOPRINT`, alors *l'ODS est désactivé*. Aucun objet ne sera créé pour cette procédure. Il est préférable, pour ne pas

produire de sorties dans une destination en particulier, d'utiliser les listes noires `ODS SELECT` et `ODS EXCLUDE` (cf. section 1.2.2).

1.1.1 La récupération des noms et labels d'objets

Un objet ODS est l'unité élémentaire des sorties produites par une procédure. Il s'agit généralement de texte mis en forme de manière tabulaire, avec lignes et colonnes ; mais les procédures graphiques produisent également des objets : chaque graphique est un objet ODS distinct.

Chaque objet ODS est caractérisé par son nom : il indique son contenu et dépend de la procédure qui l'a produit. On édite les caractéristiques des objets ODS produits avec l'instruction `ODS TRACE`.

Exemple 1.01 – Objets ODS

```
ODS TRACE ON ;
PROC CONTENTS DATA = livre.voitures ;
RUN ;
ODS TRACE OFF ;
```

Dans la fenêtre Log, on trouve des blocs correspondant aux quatre objets ODS produits. Voici les deux premiers :

```
Output Added:
-------------
Name:        Attributes
Label:       Attributes
Template:    Base.Contents.Attributes
Path:        Contents.DataSet.Attributes

Output Added:
-------------
Name:        EngineHost
Label:       Engine/Host Information
Template:    Base.Contents.EngineHost
Path:        Contents.DataSet.EngineHost
```

Les noms des deux objets créés ici sont `Attributes` et `EngineHost`. Il en va de même quelle que soit la langue de la session SAS.

1.1.2 La création de tables SAS

La création de tables SAS *via* l'ODS se distingue de toutes les autres syntaxes de l'ODS. C'est la seule à s'appuyer sur les noms d'objets pour fonctionner ; il est donc souvent nécessaire de recourir aux manipulations comme `ODS TRACE` avant de créer une table SAS *via* l'ODS. Elle ne présente pas non plus d'instruction fermante (si `ODS OUTPUT CLOSE ;` sera exécuté sans erreur, cette instruction est totalement inutile).

Quel peut être l'intérêt d'utiliser l'ODS pour créer une table SAS, alors que bien souvent la syntaxe de la procédure intègre des options OUT qui proposent la même fonctionnalité ?

- Tout d'abord la certitude de pouvoir récupérer une information dans une table SAS : parfois les options OUT ne permettent pas de créer des tables contenant l'intégralité des informations produites par une procédure. ODS OUTPUT garantit qu'on retranscrira dans une table une sortie que la procédure sait afficher dans la fenêtre Output de SAS.
- Autre intérêt, l'harmonisation de la syntaxe : là où les options OUT se retrouvent à divers endroits de la syntaxe des procédures, ODS OUTPUT s'intègre toujours de la même manière aux programmes.

```
ODS OUTPUT nomObjet = nomTableSAS ;
PROC xxx … ;
…
RUN ; /* fin implicite de l'instruction ODS */
```

Pour utiliser cette destination de l'ODS, il suffit d'ajouter avant le début de la procédure une instruction ODS OUTPUT qui lie un objet et le nom de la table SAS créée.

Attention : ODS OUTPUT s'applique à la procédure qui suit *immédiatement*. Il ne faut donc pas conserver de procédure produisant des sorties ; la procédure SORT et l'étape Data sont autorisées, mais pas une procédure avec l'option NOPRINT.

Exemple 1.02 – Création de tables SAS

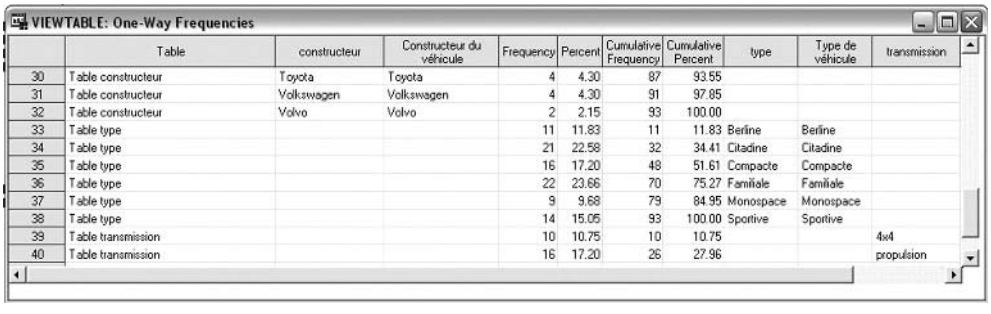
```
ODS OUTPUT nLevels = work.cardinalites oneWayFreqs = work.frequences ;
TITLE1 "Produit 2 tables SAS en plus des sorties usuelles" ;
PROC FREQ DATA = livre.voitures NLEVELS ;
  TABLE constructeur type transmission ;
RUN ;
TITLE1 "Ne produit aucune table SAS" ;
PROC FREQ DATA = livre.voitures ;
  TABLE passagers ;
RUN ;
TITLE ;
```

Les deux tables SAS créées par l'exemple 1.03 se présentent ainsi (figure 1.2 et 1.3).

VIEWTABLE: Number of Variable Levels

	Table Variable	Table Variable Label	Number of Levels
1	constructeur	Constructeur du véhicule	32
2	type	Type de véhicule	6
3	transmission	Mode de transmission	3

Figure 1.2 — La table CARDINALITES (correspond à l'objet NLEVELS)



VIEWTABLE: One-Way Frequencies

	Table	constructeur	Constructeur du véhicule	Frequency	Percent	Cumulative Frequency	Cumulative Percent	type	Type de véhicule	transmission
30	Table constructeur	Toyota	Toyota	4	4.30	87	93.55			
31	Table constructeur	Volkswagen	Volkswagen	4	4.30	91	97.85			
32	Table constructeur	Volvo	Volvo	2	2.15	93	100.00			
33	Table type			11	11.83	11	11.83	Berline	Berline	
34	Table type			21	22.58	32	34.41	Citadine	Citadine	
35	Table type			16	17.20	48	51.61	Compacte	Compacte	
36	Table type			22	23.66	70	75.27	Familiale	Familiale	
37	Table type			9	9.68	79	84.95	Monospace	Monospace	
38	Table type			14	15.05	93	100.00	Sportive	Sportive	
39	Table transmission			10	10.75	10	10.75			4x4
40	Table transmission			16	17.20	26	27.96			propulsion

Figure 1.3 — La table FREQUENCES (correspond à l'objet ONEWAYFREQS)

On remarque que dans la table FREQUENCES, les trois tableaux édités par la procédure FREQ (un par variable) sont rassemblés les uns à la suite des autres. Si on préfère obtenir des tables SAS séparées, il faut ajouter à côté du nom de l'objet, dans l'instruction `ODS OUTPUT`, l'option `MATCH_ALL` entre parenthèses. Cela fonctionne à chaque fois qu'une procédure produit plusieurs objets de noms identiques (et donc aussi avec une instruction `BY`, par exemple).

Les noms de ces tables sont générés automatiquement : si le nom proposé dans l'instruction ne se termine pas par un chiffre, il sera utilisé pour la première table produite, puis suffixé d'un 1 pour la deuxième table, puis d'un 2 pour la troisième, etc. Il est donc plus logique de proposer un nom se terminant par un nombre, de manière à ce que SAS l'incrémente à chaque nouvelle table créée.

Exemple 1.03 – Création de tables SAS : option MATCH_ALL

```
ODS OUTPUT oneWayFreqs (MATCH_ALL) = work.frequences1 ;
PROC FREQ DATA = livre.voitures ;
       TABLE constructeur type transmission ;
RUN ;
```

Ici, le programme produit trois tables appelées FREQUENCES1, FREQUENCES2 et FREQUENCES3.

1.2 LES DESTINATIONS ODS : QUELS FORMATS DE FICHIERS CRÉER ET COMMENT ?

1.2.1 Que faire (et ne pas faire) avec l'ODS ?

On appelle *destination ODS* un type de fichier que SAS sait produire :

- pour les sorties SAS dans la fenêtre Output, la destination est LISTING ;
- pour les pages web, la destination est HTML ;
- pour les documents Word[1], la destination est RTF ;
- pour les documents PDF, la destination est PDF ;
- pour les fichiers XML, la destination est MARKUP ou TAGSETS ;
- pour les classeurs Excel, la destination est TAGSETS.EXCELXP (ou, pour Excel 2000 et Excel 2002, TAGSETS.MSOFFICE2K) ;
- pour les tables SAS, la destination est OUTPUT.

Les destinations HTML et TAGSETS ne sont pas prévues pour être imprimées ; la notion de page n'y est donc pas très développée.

La destination RTF produit du texte à insérer dans un document final, mais ne permet pas toujours de produire un document directement diffusable sans aucune modification ; alors que la destination PDF est justement pensée pour des documents non modifiables, à exploiter tels quels. On retrouvera donc une notion de longueur de page très maîtrisée en PDF, mais pas d'équivalent en RTF (la version 9.2 de SAS propose une destination appelée « RTF mesuré » expérimentale, permettant de mieux contrôler l'emplacement des tableaux produits dans une page).

Enfin, la destination TAGSETS.EXCELXP a vocation à produire des classeurs Excel ; un certain nombre de fonctionnalités d'Excel sont supportées (filtres, entêtes et pieds de pages, volets figés) mais pour la mise en forme des tableaux, polices, bordures et trames ne sont pas toujours malléables ; il en va de même pour TAGSETS.MSOFFICE2K qui est un format HTML *légèrement* adapté à Excel.

1.2.2 L'exclusion et l'inclusion : la gestion de « listes noires »

Pour chacune de ses destinations, l'ODS gère une liste noire d'objets qui ne seront pas traités. Par défaut, ces listes sont vides. On peut les alimenter ou les vider à l'aide des instructions `ODS SELECT` et `ODS EXCLUDE`. La première indique quels sont les seuls objets à traiter, la seconde quels sont les objets à ne pas traiter.

Ces instructions peuvent être ciblées sur une destination en particulier : `ODS RTF EXCLUDE` ne concerne que la liste noire pour la destination RTF. Si on omet la desti-

1. Par « document Word », on désignera dans tout cet ouvrage les fichiers susceptibles d'être modifiés par un logiciel de traitement de texte, qu'il s'agisse ou non de Microsoft Word ; cf. section 1.2.4 pour plus de détails.

nation, toutes sont concernées, sauf OUTPUT (création de tables SAS) qui est indépendante.

```
ODS <destination> EXCLUDE|SELECT nomObjet|ALL|NONE ;
PROC xxx … ;
…
RUN ; /* réinitialisation de la liste noire */
```

Si on exécute `ODS EXCLUDE ALL ;` alors plus aucune sortie de la procédure qui suit n'est envoyée aux différentes destinations de l'ODS. Cette instruction est préférable à l'option `NOPRINT` d'une procédure, car l'ODS reste actif pour créer des tables SAS via ODS OUTPUT.

Ces listes noires sont vidées par défaut à la fin de chaque procédure si elles ne concernent que certains objets. Pour garder une liste noire en l'état d'une procédure sur l'autre, il faut ajouter l'option PERSIST entre parenthèses à côté des noms d'objets concernés. Mais si la liste noire concerne tous les objets (ODS EXCLUDE ALL), elle ne sera vidée qu'à l'exécution de l'instruction ODS SELECT ALL.

Exemple 1.04 – Objets ODS : gestion de listes d'exclusion

```
ODS LISTING EXCLUDE attributes (PERSIST)
                    engineHost ;
TITLE1 "Sortie sans ATTRIBUTES ni ENGINEHOST" ;
PROC CONTENTS DATA = livre.voitures ;
RUN ;
TITLE1 "Sortie sans ATTRIBUTES mais avec ENGINEHOST" ;
PROC CONTENTS DATA = livre.voitures ;
RUN ;
ODS SELECT ALL ; /* purge toutes les listes noires */
TITLE1 "Sortie normale" ;
PROC CONTENTS DATA = livre.voitures ;
RUN ;
TITLE ;
```

1.2.3 La syntaxe de base

On l'a dit, chaque destination ODS correspond à un type de fichier différent. Hormis OUTPUT, elles fonctionnent toutes sur la même logique : une instruction fermante répond à une instruction ouvrante. Elles encadrent l'ensemble des traitements (procédures et étapes Data) dont le résultat doit être inclus dans le document produit.

```
ODS destination FILE = "fichier à créer"
             < STYLE = charteGraphique >
           < NEWFILE = PAGE | BYGROUP | TABLE | PROC >
              < autres options > ;
PROC xxx … ;
RUN ;
/* éventuellement, autres procédures et/ou étapes Data */
ODS destination CLOSE ;
```

L'option FILE indique l'emplacement du fichier à créer. Sur Windows ou Unix, on indiquera dans cette option, en général, le répertoire et le nom du fichier entre guillemets. Sur MVS, on aura associé par l'instruction FILENAME un nom logique (*fileref*) à un emplacement disponible, existant (avec l'option DISP=OLD) ou non (DISP=NEW) au préalable, et on indiquera dans FILE le nom du *fileref*, sans guillemets cette fois. Les utilisateurs maîtrisant bien les arcanes de MVS pourront enrichir le FILENAME d'options pour indiquer la longueur d'enregistrement (LRECL) ou le nombre de blocs (BLKSIZE).

```
FILENAME sortie "user.temp.ods" DISP = NEW|OLD ;
ODS destination FILE = sortie … ;
```

L'option STYLE sera détaillée au chapitre 5, où l'on verra comment créer sa propre charte graphique. On peut lister l'ensemble des styles possibles avec la procédure Template (d'autres manipulations pour obtenir le même résultat seront proposées aux chapitres 5 et 7). Des styles possibles sont entre autres BEIGE, BROWN, BARRETTSBLUE, PRINTER, RTF (tous disponibles dès la version 8), ASTRONOMY, BANKER, JOURNAL, SEASIDE, STATISTICAL.

```
PROC TEMPLATE ;
   LIST styles ;
RUN ;
```

L'option NEWFILE permet de créer, en une seule fois, plusieurs documents de même type. Derrière cette option, un évènement est précisé : à chaque fois qu'il se produit, l'ODS ferme le document courant et en commence un nouveau. Les évènements sont PAGE (saut de page), BYGROUP (changement de valeur d'une variable citée dans un BY), TABLE (création d'un nouvel objet ODS par la procédure) ou encore PROC (début d'une procédure). Le nom des fichiers générés suit la même logique que précédemment : si le nom proposé dans l'option FILE ne se termine pas par un chiffre, il sera utilisé pour le premier fichier produit, puis suffixé d'un 1 pour le deuxième fichier, puis d'un 2 pour le troisième, etc. Il est donc plus logique de proposer un nom se terminant par un nombre, de manière à ce que SAS l'incrémente à chaque nouveau fichier créé.

Attention au cas de la procédure PRINT : dans la version 8 de SAS, son résultat est considéré comme un seul objet, même s'il est entrecoupé de sauts dus à une instruction BY. Donc la combinaison PROC PRINT…BY + NEWFILE=BYGROUP ne fonctionne pas dans cette version, puisqu'on ne change pas d'objet d'un bloc BY à l'autre : on produit au final un seul document. Ce problème est résolu dans SAS 9, l'alternative consistant à utiliser PROC PRINT…BY…PAGEBY et NEWFILE=PAGE (on change de page avec chaque bloc BY et on change de document à chaque saut de page).

Dans toutes les destinations sauf OUTPUT, les procédures (à l'exception notable de PRINT, TABULATE et REPORT) ont leurs sorties précédées d'une ligne de titre indiquant quelle procédure a produit ces résultats. On peut éliminer cette ligne d'information par l'instruction ODS NOPROCTITLE ; et l'activer à nouveau par ODS PROCTITLE ;. Il s'agit d'une instruction à effet « permanent », dont l'action dure jusqu'à la fermeture de la session SAS.

Si la base de la syntaxe est commune, des options et des précautions sont spécifiques à chaque destination. La suite de cette partie les passe en revue.

1.2.4 L'affichage des sorties en fonction du logiciel utilisé

En raison de la prééminence des logiciels Microsoft sur les ordinateurs actuels, SAS construit ses sorties de manière à obtenir un rendu impeccable sur Word, Excel ou Internet Explorer. Cette courte partie propose de lister les écarts qui peuvent exister selon le logiciel utilisé, pour les trois destinations ODS sensibles : HTML, RTF et TAGSETS.EXCELXP.

- La destination HTML – Microsoft Internet Explorer (version 6.0), Mozilla Firefox (version 2.0) et Opera (version 9.27) rendent les pages produites par ODS HTML de manière légèrement différente. En particulier, les bordures des tableaux présentent une esthétique variable, conséquence de la prise en compte hétérogène des attributs « largeur de la bordure de cellule ».
- La destination RTF – Les documents RTF s'affichent de manière quasiment identique dans Microsoft Word (version 2003) et dans OpenOffice Writer (version 2.4). Cependant, si on veut personnaliser la sortie avec du code RTF comme dans l'exemple 2.07, le résultat n'est pas identique.
- La destination TAGSETS.EXCELXP – La comparaison entre OpenOffice Calc (version 2.4) et Microsoft Excel (version 2003) tourne court : la destination TAGSETS.EXCELXP est expressément destinée au logiciel de Microsoft. Si on essaye d'ouvrir avec Calc le fichier produit par l'exemple 1.12, on est accueilli par l'écran d'import d'un fichier texte, preuve que le XML pour MS Excel n'est pas interprété correctement.

> Dans toute la suite de ce livre, nous utiliserons donc les logiciels Microsoft pour afficher le rendu des programmes SAS. Toute utilisation des exemples dans des logiciels autres ne donnera pas forcément un résultat cohérent.

1.2.5 L'ouverture simultanée de plusieurs destinations ODS

Il est tout à fait possible d'ouvrir simultanément plusieurs destinations ODS. Il est même possible d'ouvrir plusieurs fois la même destination ODS pour créer deux fichiers distincts, par exemple en jouant sur des options différentes.

Pour distinguer plusieurs destinations ODS identiques ouvertes simultanément, on nommera l'ODS à l'aide d'une des deux syntaxes suivantes.

```
ODS destination (numéro) options ;
ODS destination (NAME="nom libre") options ;
```

On utilisera ensuite la même convention pour fermer (instruction `ODS … CLOSE`) la destination concernée, en précisant son numéro ou son nom entre parenthèses.

1.3 LE DÉTAIL DES DESTINATIONS ODS COURANTES

1.3.1 La création de pages web

La production de pages web est intéressante en soi, car elle permet de diffuser des informations sur des Intranets et des sites Internet d'entreprises. Elle peut être utilisée en réponse à des demandes effectuées *via* un formulaire et le module SAS/IntRnet en réponse à un formulaire. Mais une page HTML peut également être ouverte par les logiciels de la suite Microsoft Office (en particulier Word et Excel) depuis leurs versions 97.

En outre, la relative simplicité du langage HTML permet de personnaliser aisément les sorties, en intégrant des commandes dans ce langage dans des titres, des pieds de page, des formats (cf. section 2.2.2).

La logique d'une page HTML

Une page HTML est constituée des éléments suivants :

- un en-tête qui indique le logiciel ayant généré le fichier, l'auteur, divers mots-clés, ainsi que des éléments de mise en forme générale (la feuille de style, qui peut être également « détachée » dans un fichier CSS séparé) ;
- le corps (BODY) de la page HTML qui est composé d'un mélange de texte à afficher et de balises (instructions de mise en forme en langage HTML) ;
- certaines balises font référence à d'autres fichiers (autres pages HTML, images, autres documents) à travers un système de liens (dits liens hypertextes).

La logique des balises est la suivante : si on veut écrire le texte « Reporting avec SAS » en gras, on l'encadrera des balises `<B>` (pour Bold, c'est-à-dire gras) et `</B>` qui vient annuler l'effet de la première : le texte suivant `</B>` ne sera pas en gras. On écrira donc `<B>Reporting avec SAS</B>`.

La logique d'un lien est d'indiquer comment aller du fichier courant au fichier auquel on se réfère. Le chemin proposé peut être absolu (on indique l'adresse physique du fichier, par exemple `c:\temp\mon_image.gif`) ou, de préférence, relatif (on indique comment aller du fichier en cours au fichier lié, par exemple `mon_image.gif` si le fichier est dans le même répertoire, `/temp/mon_image.gif` si le fichier d'où on part est situé dans le répertoire `c:\`, etc.).

Un fichier simple, une table des matières et une frame

Avec la syntaxe de base, on crée une seule page web, celle qui est citée dans l'option `FILE`. Mais il est également possible de créer plusieurs pages liées les unes aux autres : pour cela, deux options complémentaires, `CONTENTS` et `FRAME`, permettent de désigner respectivement une page web servant de sommaire aux sorties (ce sera une suite de liens vers les différents éléments de la page HTML de l'option `FILE`) et une page web qui servira de contenant, mettant sommaire et sortie principale côte à côte.

```
ODS HTML FILE = "fichier à créer"      < STYLE = charteGraphique >
    < NEWFILE = PAGE | BYGROUP | TABLE | PROC >
   < CONTENTS = "fichier sommaire" >   < FRAME = "fichier contenant" >
       < PATH = "répertoire commun aux 3 fichiers" (URL=NONE) > ;
…
ODS HTML CLOSE ;
```

Comme on produit plusieurs fichiers, on les stocke généralement dans un même répertoire ; il est commode d'indiquer celui-ci dans l'option `PATH`, et de ne donner que les noms seuls des fichiers dans les options `FILE`, `CONTENTS` et `FRAME`. La précision `URL=NONE` à côté du répertoire `PATH` permet de construire des chemins relatifs dans les liens entre fichiers (par défaut, SAS crée des liens avec des chemins absolus, ce qui pose des problèmes quand on déplace les pages web produites).

Exemple 1.05 – Création de pages web : options PATH, CONTENTS, FRAME

```
ODS HTML PATH = "c:\temp" (URL=NONE) FILE = "sortie_seule.htm"
     CONTENTS = "sommaire.htm"      FRAME = "sortie_complete.htm" ;
PROC MEANS DATA = livre.voitures MEAN MEDIAN ;
  VAR conso_ville conso_auto ;
  CLASS type transmission ; WAYS 0 1 ;
RUN ;
ODS HTML CLOSE ;
```

La figure 1.4 présente SORTIE_COMPLETE.HTM, avec sur la gauche la page SOMMAIRE.HTM et sur la droite la page SORTIE_SEULE.HTM.

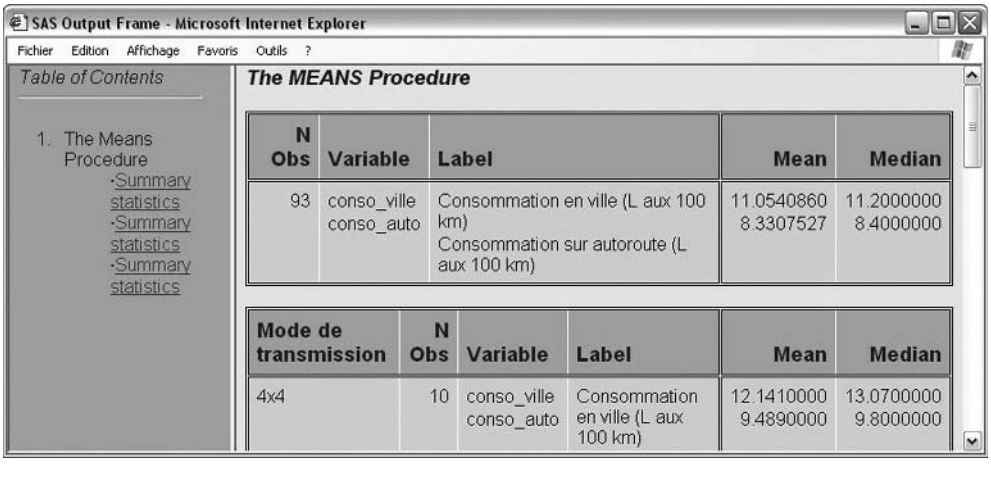


Figure 1.4 — Résultat de l'exemple 1.05, le fichier SORTIE_COMPLETE.HTM

Les graphiques

Un point problématique avec la production de pages web est l'insertion d'images. Les fichiers HTML *ne contiennent pas d'images*, mais seulement des liens vers des

fichiers images séparés. On précise donc généralement un répertoire (déjà existant) de stockage pour ces images (option GPATH) ; en version 9, si on omet cette option les graphiques vont dans le même répertoire que le fichier principal (options FILE ou PATH). Comme pour l'option PATH, il est recommandé de préciser après le répertoire de GPATH un lien relatif entre le répertoire du fichier principal et le répertoire d'images : on utilisera encore la sous-option URL.

```
ODS HTML FILE = "fichier à créer"
     < STYLE = charteGraphique >
     < GPATH = "répertoire des images" (URL=NONE|URL="lien relatif") > ;
```

Cette précision du répertoire de stockage des images n'est pas utile si on utilise les drivers graphiques ActiveX ou Java : les « images » sont alors incluses dans le fichier principal. Pour plus de précisions sur les graphiques et les drivers, voir le chapitre 3.

Exemple 1.06 – Création de pages web : intégration de graphiques

```
ODS HTML FILE = "c:\temp\sortie avec graphiques.htm"
        GPATH = "c:\temp\img\" (URL="img/") ;
PROC GCHART DATA = livre.voitures ;
   VBAR type / SUBGROUP = transmission ;
RUN ; QUIT ;
ODS HTML CLOSE ;
```

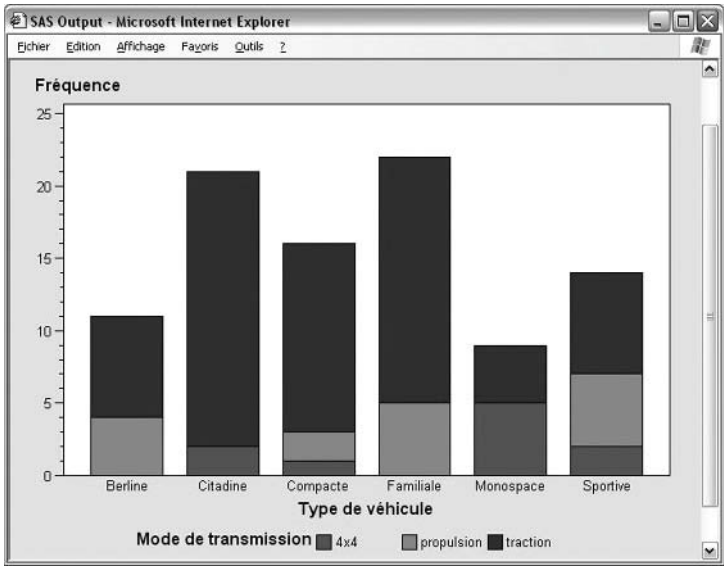


Figure 1.5 — Résultat de l'exemple 1.06, le fichier SORTIE AVEC GRAPHIQUES.HTM

La création de fichiers HTML sur MVS

Sur MVS, il convient d'ajouter systématiquement deux options pour assurer la bonne lecture sous Windows des fichiers HTML produits par SAS : `RS=NONE` et `TRANTAB=ASCII`.

```
ODS HTML FILE=sortie TRANTAB=ASCII RS=NONE < autres options > ;
```

1.3.2 La création de documents Word

La destination RTF (*Rich Text Format*) permet de produire des documents destinés aux logiciels de traitement de texte. Il est à noter que ces documents contiennent de nombreuses instructions de mise en page, mais ne permettent pas de prévoir l'emplacement des sauts de page (au contraire de la destination PDF décrite plus loin).

Les versions à venir de SAS contiendront des destinations de RTF dit « mesuré » (*measured*), c'est-à-dire où le texte à mettre en page est dimensionné en longueur avant d'être intégré au document, permettant ainsi de prévoir et de contrôler l'emplacement des sauts de page.

Le fichier de base

Les principales options de la destination RTF concernent le découpage du document en colonnes (l'option `COLUMNS` vaut 1 par défaut), l'insertion de sauts de pages entre les sorties des différentes procédures (attention aux conflits avec les en-têtes et pieds de page dans le cas où on inhiberait les sauts de page avec `STARTPAGE=NO`). À noter que si on désire insérer un saut de page entre deux procédures, on peut exécuter l'instruction `ODS RTF STARTPAGE=NOW;` à l'endroit voulu.

L'option KEEPN empêche un tableau d'être à cheval sur deux pages, quand c'est possible.

```
ODS RTF FILE = "fichier à créer" < STYLE = charteGraphique >
    < NEWFILE = PAGE | BYGROUP | TABLE | PROC >
    < COLUMNS = 1|2|3 > < STARTPAGE = NO > < KEEPN > < BODYTITLE > ;
PROC xxx … ;
…
RUN ;
/* éventuellement, autres procédures et/ou étapes Data */
< ODS RTF STARTPAGE = NOW ; >
/* éventuellement, autres procédures et/ou étapes Data */
ODS RTF CLOSE ;
```

Par défaut, les titres et pieds de page SAS (instructions `TITLE1` à `TITLE10` et `FOOTNOTE1` à `FOOTNOTE10`) sont intégrés aux en-têtes et pieds de pages du document Word. Ils sont ainsi répétés automatiquement d'une page sur l'autre. De même pour le numéro de page et la date du jour. On peut les intégrer au corps du document avec l'option `BODYTITLE` ; les répétitions automatiques de titres sur des sorties étalées sur plusieurs pages sont alors perdues.

Exemple 1.07 – Création de documents Word

```
ODS RTF FILE = "c:\temp\sortie_rtf.doc" STARTPAGE = NO ;
TITLE1 "Titre de la procédure FREQ" ;
PROC FREQ DATA = livre.voitures ;
   TABLE transmission ;
RUN ;
TITLE1 "Titre de la procédure MEANS, qui ne sera pas affiché" ;
PROC MEANS DATA = livre.voitures MEAN MEDIAN ;
   VAR conso_: ;
RUN ;
TITLE ;
ODS RTF CLOSE ;
```

1

Titre de la procédure FREQ

The FREQ Procedure

Mode de transmission				
transmission	Frequency	Percent	Cumulative Frequency	Cumulative Percent
4x4	10	10.75	10	10.75
propulsion	16	17.20	26	27.96
traction	67	72.04	93	100.00

The MEANS Procedure

Variable	Label	Mean	Median	Variable	Label	Mean	Median
conso_ville	Consommation en ville (L aux 100 km)	11.0540860	11.2000000	conso_auto	Consommation sur autoroute (L aux 100 km)	8.3307527	8.4000000

Figure 1.6 — Résultat de l'exemple 1.07, le fichier SORTIE_RTF.DOC

La table des matières

Il est possible (en SAS version 9) d'intégrer au début du document Word une table des matières. Plus exactement, SAS génère le texte invisible (les signets) pour signaler des entrées de table des matières, et commence le corps du document par une instruction de création d'une table des matières qui doit ensuite être activée manuellement.

Côté SAS, on ajoute à l'instruction `ODS RTF` l'option `CONTENTS`. Côté Word, il faut mettre à jour les champs du document (enchaîner la combinaison de touches « `CTRL+A` » et la touche `F9`). La table des matières apparaît alors au début du document généré.

Exemple 1.08 – Création de documents Word avec table des matières

```
ODS RTF FILE = "c:\temp\sortie_rtf.doc" CONTENTS ;
PROC FREQ DATA = livre.voitures ;
```

```
        TABLE transmission ;
RUN ;
PROC MEANS DATA = livre.voitures MEAN MEDIAN ;
        VAR conso_: ;
RUN ;
ODS RTF CLOSE ;
```

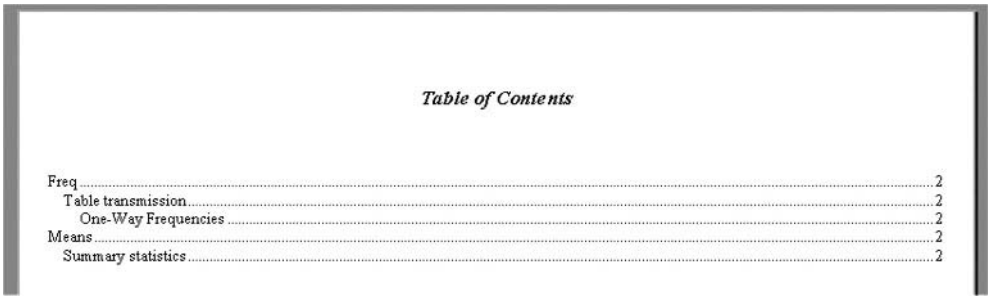


Figure 1.7 — Résultat de l'exemple 1.08, le fichier SORTIE_RTF.DOC

La création de fichiers RTF sur MVS

Par rapport au cas du HTML, seule l'option `TRANTAB=ASCII` est indispensable quand on utilise la destination RTF sur une machine MVS.

```
ODS RTF FILE = sortie TRANTAB = ASCII < autres options > ;
```

1.3.3 La création de documents PDF

La destination PDF (*Portable Document Format*) fait partie d'une famille de destinations qui comprend PRINTER (l'envoi d'un document à l'imprimante par défaut, risqué car automatique) et POSTSCRIPT (la création de documents imprimés sur une imprimante virtuelle). Le but poursuivi par ce type de format est double : d'abord produire un document dont l'impression est sans surprise (il est vu à l'écran exactement à l'identique de ce qu'il sera à l'impression) et également que ce document soit non modifiable. Des options de cryptage sont ajoutées à partir de SAS version 9.2.

Le fonctionnement de la destination PDF se fait indépendamment du fait que l'ordinateur exécutant le programme possède une licence pour le logiciel Acrobat Distiller ou un équivalent.

Le fichier de base

Les principales options de la destination ODS PDF rejoignent celles déjà vues pour HTML et RTF : `FILE`, `STYLE`, `NEWFILE` sont communes. Par défaut, des signets sont insérés pour chaque objet du document. L'option `CONTENTS` permet d'insérer (sans autre intervention cette fois) une table des matières en début de document, tandis que `BOOKMARKLIST=HIDE` permet de masquer le navigateur de signets (la partie gauche

de l'affichage dans Acrobat Reader). Enfin, l'option NOTOC annule la génération de signets. L'option UNIFORM harmonise la largeur d'un tableau éclaté sur plusieurs pages.

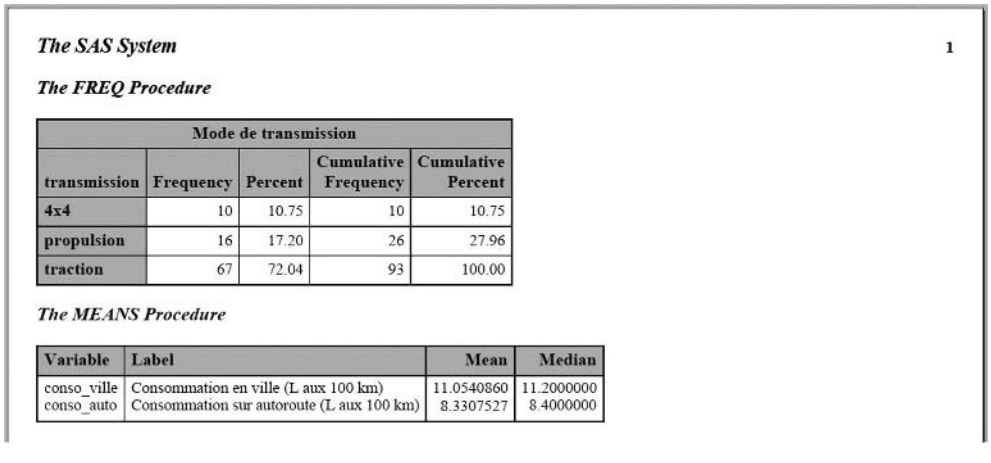
On retrouve également l'option STARTPAGE=NO déjà vue avec ODS RTF ; on peut également forcer l'insertion de sauts de page avec des instructions ODS PDF STARTPAGE=NOW ; entre deux procédures.

```
ODS PDF FILE = "fichier à créer"
    < STARTPAGE = NO > < STYLE = charteGraphique >
    < NEWFILE = PAGE | BYGROUP | TABLE | PROC > < UNIFORM >
    < CONTENTS > < NOTOC > < BOOKMARKLIST = HIDE >
    < AUTHOR = "…" SUBJECT = "…" TITLE = "…" KEYWORDS = "…" > ;
…
ODS PDF CLOSE ;
```

On peut enfin proposer des valeurs pour les propriétés du document produit : nom de l'auteur, mots-clés, titre et sujet, *via* les mots-clés (respectivement) AUTHOR, KEYWORDS, TITLE et SUBJECT. Aucun de ces textes ne s'affiche à l'écran dans le corps du document, mais uniquement dans le menu de propriétés et dans l'explorateur Windows s'il est configuré pour cela.

Exemple 1.09 – Création de documents PDF

```
ODS PDF FILE = "c:\temp\sortie.pdf" STARTPAGE = NO
      AUTHOR = "Olivier Decourt" TITLE = "Démonstration ODS PDF" ;
PROC FREQ DATA = livre.voitures ;
       TABLE transmission ;
RUN ;
PROC MEANS DATA = livre.voitures MEAN MEDIAN ;
       VAR conso_: ;
RUN ;
ODS PDF CLOSE ;
```



The SAS System 1

The FREQ Procedure

Mode de transmission				
transmission	Frequency	Percent	Cumulative Frequency	Cumulative Percent
4x4	10	10.75	10	10.75
propulsion	16	17.20	26	27.96
traction	67	72.04	93	100.00

The MEANS Procedure

Variable	Label	Mean	Median
conso_ville	Consommation en ville (L aux 100 km)	11.0540860	11.2000000
conso_auto	Consommation sur autoroute (L aux 100 km)	8.3307527	8.4000000

Figure 1.8 — Résultat de l'exemple 1.09, le fichier SORTIE.PDF

Les propriétés du document vu à la figure 1.8 sont consultables dans le menu FICHIER > PROPRIETES DU DOCUMENT d'Adobe Acrobat Reader : on les voit sur la figure 1.9.

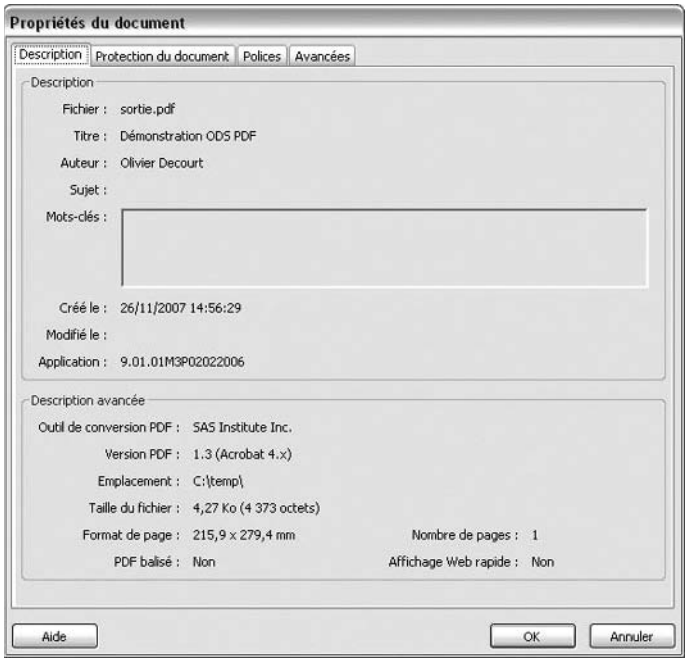


Figure 1.9 — Résultat de l'exemple 1.09, les propriétés du fichier SORTIE.PDF

La gestion des polices

Les documents PDF incluent en leur sein la définition des polices qui y sont utilisées, au contraire d'un document Word ou une page web, où l'affichage et la recherche des polices sont laissés au logiciel de visualisation. Pour que SAS puisse « embarquer » la définition d'une police « spéciale » (qui ne serait pas déjà inscrite par défaut), il faut la déclarer *via* la procédure FONTREG.

```
PROC FONTREG MODE=ADD ;
   FONTPATH "répertoire de stockage des polices" ;
RUN ;
```

Ces polices sont alors stockées dans le registre de SAS, c'est-à-dire sur la machine qui exécute la session SAS. Il est donc normalement inutile de relancer ce programme à chaque exécution de la destination PDF sur un même poste.

Exemple 1.10 – Création de documents PDF intégrant des polices Windows

```
PROC FONTREG MODE = ADD ;
   FONTPATH "c:\windows\fonts" ;
RUN ;
```

```
ODS PDF FILE = "c:\temp\sortie avec polices Windows.pdf" ;
TITLE1 FONT="Jokerman" "Ceci est une police JOKERMAN" ;
PROC FREQ DATA = livre.voitures ;
   TABLE transmission ;
RUN ;
TITLE ;
ODS PDF CLOSE ;
```

Ceci est une police JOKERMAN

The FREQ Procedure

Mode de transmission				
transmission	Frequency	Percent	Cumulative Frequency	Cumulative Percent
4x4	10	10.75	10	10.75
propulsion	16	17.20	26	27.96
traction	67	72.04	93	100.00

Figure 1.10 — Résultat de l'exemple 1.10, le fichier SORTIE AVEC POLICES.PDF

1.3.4 La création de classeurs Excel

Les versions les plus récentes d'Excel (2002 et supérieures) savent lire, en plus des formats XLS et HTML, un format spécifique de fichiers XML (selon une norme développée par Microsoft). SAS peut, depuis la version 9, produire ce genre de fichiers. On utilise pour cela encore l'ODS mais avec une destination appelée TAGSETS.EXCELXP : le fichier produit est plus spécifique à Excel que la destination HTML, et permet de créer de vrais classeurs multi-onglets.

La syntaxe de cette destination n'est pas vraiment harmonisée avec le reste de l'ODS. Il est également conseillé d'aller régulièrement sur le site de l'éditeur SAS pour télécharger une nouvelle version du jeu de balises associé à cette destination : il s'agit d'un simple programme SAS à exécuter. Cela permet d'utiliser de nouvelles options et d'obtenir des résultats plus robustes.

Le fichier de base

On retrouve les options usuelles de l'ODS, comme `FILE`, `STYLE`, `NEWFILE`. Les autres options sont à préciser dans une syntaxe plus inhabituelle, et ces options sont assez nombreuses.

```
ODS TAGSETS.EXCELXP FILE = "chemin et nom du fichier créé"
                 < STYLE = charteGraphique >
               < NEWFILE = PAGE | BYGROUP | TABLE | PROC >
                 < OPTIONS (nomOption1="valeur1"
```

```
                                < nomOption2="valeur2" >
                                 ) > ;
…
ODS TAGSETS.EXCELXP CLOSE ;
```

Tableau 1.1 — Options les plus courantes de la destination ODS TAGSETS.EXCELXP (version 1.86 ; 15/04/2008)

Option	Valeurs possibles (*par défaut*)	Usage
Orientation	*PORTRAIT*, LANDSCAPE	Orientation à l'impression.
Zoom	*100*, un nombre	Valeur du zoom d'affichage.
Scale	*100*, un nombre	Valeur du zoom d'impression.
BlackAndWhite	*NO*, YES	Imprime en noir et blanc.
DraftQuality	*NO*, YES	Imprime en qualité brouillon.
Gridlines	*NO*, YES	Affiche le quadrillage automatique à l'impression.
FitToPage	*NO*, YES	Ajuste le zoom d'impression pour faire tenir la feuille dans x pages ; cette option ne prend effet que conjointement avec PAGES_FITWIDTH et PAGES_FITHEIGHT (par défaut, faire tenir sur une seule page toute la feuille).
Pages_FitWidth Pages_FitHeight	*1*, un nombre	Ajuste la largeur (respectivement la hauteur) de chaque feuille sur x pages à l'impression.
Embedded_Titles	*NO*, YES	YES : les Footnotes SAS sont considérés comme pieds de page Excel et pas affichés directement dans la feuille de calcul[1].
Suppress_Bylines	*NO*, YES	YES : les lignes de séparation de blocs BY n'apparaissent pas dans la feuille. Ne pas utiliser l'option NOBYLINE de SAS car conflit avec cette destination de l'ODS.
Frozen_Headers	*NO*, YES	La ligne d'en-tête des colonnes est toujours visible à la navigation.
AutoFilter	*NONE*, ALL, un intervalle (3-5 par exemple)	Crée des filtres automatiques sur toutes les colonnes ou sur certaines.
Row_Repeat Col_Repeat	*NONE*, un nombre	À l'impression, répète à chaque page les x premières lignes (respectivement colonnes) de la feuille.

Option	Valeurs possibles (*par défaut*)	Usage
Hidden_Columns	*NONE*, un nombre, un intervalle, une liste de numéros	Masque les colonnes dont les numéros sont indiqués (exemples : 3-5 ou 4,6, ou encore 3,4,9-11).
Formulas	*YES*, NO	Accepte les formules dans les valeurs proposées depuis SAS pour les cellules Excel.
Convert_ Percentages	*YES*, NO	Yes : les pourcentages (au format PERCENT dans SAS) sont convertis en pourcentages Excel.
Sheet_Interval	*TABLE*, PAGE, BYGROUP, PROC, NONE	Spécifie quand on change de feuille dans le classeur (à chaque tableau, saut de page, groupe BY, procédure, jamais).
Sheet_Name	*NONE*, texte	Nom de base des feuilles (agrémenté d'un compteur automatique).
Sheet_Label	*NONE*, texte	Permet de remplacer la fin du nom automatique de la feuille[2].
Contents	*NO*, YES	Crée une feuille contenant une table des matières des autres feuilles.
Index	*NO*, YES	Crée une feuille contenant un index des autres feuilles.

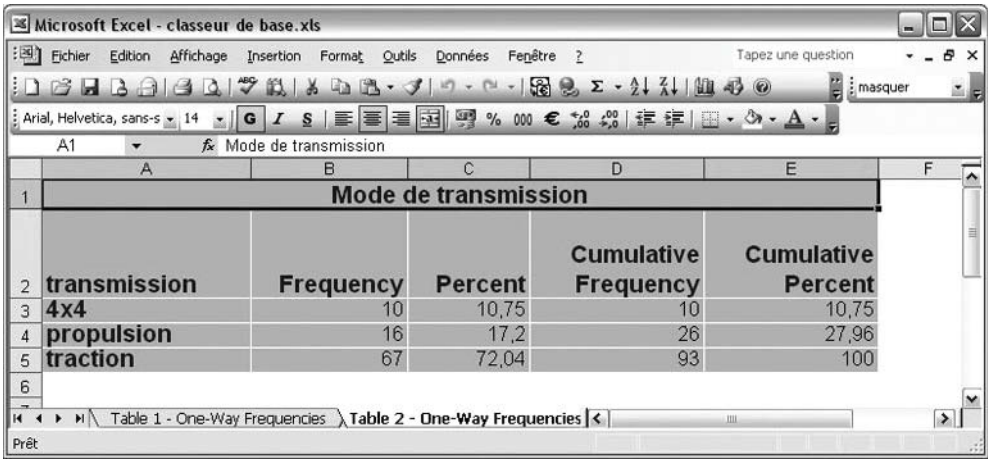
1. On peut insérer des éléments spéciaux dans les Footnotes tels que : `%NRSTR(&p)` le n° de page ; `%NRSTR(&t)` le nombre total de pages ; `%NRSTR(&h)` l'heure de création du document ; `%NRSTR(&j)` le jour de création du document ; `%NRSTR(&a)` le nom de l'onglet courant ; `%NSTR(&n)` le nom du classeur Excel ; `%NRSTR(&z)` le répertoire de stockage du classeur.
2. Les noms automatiques sont : « Proc *nbDeProcs* – label » ; « Page *nbDePages* – label » ; « Table nbDeFeuilles – label » ; « By *nbDeFeuillesParBy* – label ».

Selon la version du jeu de balises que vous utilisez, toutes ces options ne seront pas disponibles. Celles-ci sont présentes dans la version 1.86 (la version s'affiche dans la Log lors de l'exécution d'un programme faisant appel à cette destination ODS).

Exemple 1.11 – Création d'un classeur Excel

```
ODS TAGSETS.EXCELXP FILE = "c:\temp\classeur de base.xls" ;
PROC FREQ DATA = livre.voitures ;
   TABLE constructeur transmission ;
RUN ;
ODS TAGSETS.EXCELXP CLOSE ;
```

Par défaut, chaque tableau produit par l'ODS est stocké dans une feuille différente du même classeur Excel (figure 1.11).



Microsoft Excel - classeur de base.xls

Mode de transmission

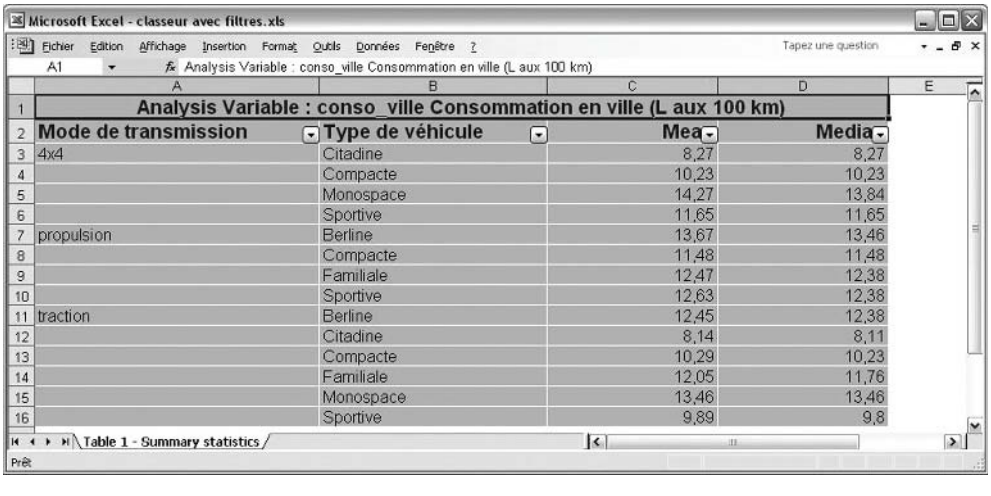
transmission	Frequency	Percent	Cumulative Frequency	Cumulative Percent
4x4	10	10,75	10	10,75
propulsion	16	17,2	26	27,96
traction	67	72,04	93	100

Figure 1.11 — Résultat de l'exemple 1.11, le fichier CLASSEUR DE BASE.XLS

Exemple 1.12 – Création d'un classeur Excel avec options

```
ODS TAGSETS.EXCELXP FILE = "c:\temp\classeur avec filtres.xls"
                OPTIONS (AUTOFILTER="ALL") ;
PROC MEANS DATA = livre.voitures MEAN MEDIAN MAXDEC=2 NONOBS ;
   VAR conso_ville ;
   CLASS transmission type ;
RUN ;
ODS TAGSETS.EXCELXP CLOSE ;
```

Chaque colonne du tableau produit est affectée d'un filtre automatique (figure 1.12).



Microsoft Excel - classeur avec filtres.xls

Analysis Variable : conso_ville Consommation en ville (L aux 100 km)

Mode de transmission	Type de véhicule	Mea	Media
4x4	Citadine	8,27	8,27
	Compacte	10,23	10,23
	Monospace	14,27	13,84
	Sportive	11,65	11,65
propulsion	Berline	13,67	13,46
	Compacte	11,48	11,48
	Familiale	12,47	12,38
	Sportive	12,63	12,38
traction	Berline	12,45	12,38
	Citadine	8,14	8,11
	Compacte	10,29	10,23
	Familiale	12,05	11,76
	Monospace	13,46	13,46
	Sportive	9,89	9,8

Figure 1.12 — Résultat de l'exemple 1.12, le fichier CLASSEUR AVEC FILTRES.XLS

1.4 L'ASPECT DU DOCUMENT PRODUIT PAR L'ODS

Plusieurs options système interagissent avec l'ODS. Certaines touchent des en-têtes systématiques (date, numéro de page), d'autres des niveaux de titres automatiques (les lignes de séparations produites par une instruction BY), d'autres encore la mise en page générale des sorties (portrait ou paysage, taille de la feuille de papier, centrage) et enfin des instructions et options permettent de personnaliser les entrées de table des matières que savent produire les destinations HTML, RTF et PDF.

1.4.1 Les en-têtes automatiques

Par défaut, les sorties SAS sont agrémentées de deux éléments automatiques en plus des titres : le numéro de page et la date (et l'heure) de démarrage de la session SAS. Ces informations ne sont pas reportées dans les destinations ODS HTML et OUTPUT, dans lesquelles la notion de page n'existe pas. Pour les autres destinations, on peut les supprimer avec les options NODATE et NONUMBER.

```
OPTION NODATE NONUMBER ;
OPTION PAGENO = 1 ;
```

En cas de besoin, on peut réinitialiser entre deux procédures les numéros de page avec l'option PAGENO. Celle-ci peut également être utile pour faire démarrer la numérotation des pages à une certaine valeur, afin de pouvoir imprimer le document à la suite d'un existant.

1.4.2 Les sorties par blocs

L'utilisation dans une procédure d'une instruction BY permet de répéter et d'éclater les traitements par blocs d'observations possédant les mêmes valeurs pour les variables de l'instruction BY.

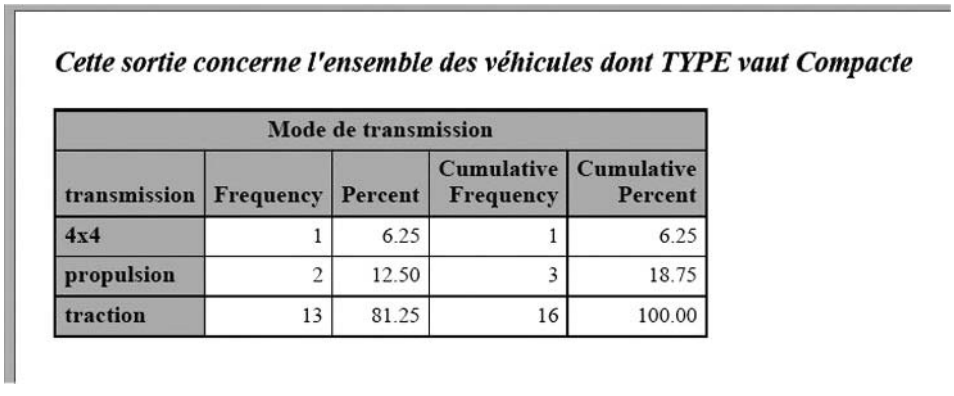
À chaque début de bloc BY, une ligne de titre automatique est générée, qui indique les caractéristiques du sous-ensemble d'observations traité. On peut éliminer cette édition automatique avec l'option système NOBYLINE, et la réactiver avec BYLINE.

En remplacement de ces titres automatiques, on peut personnaliser des lignes de titre SAS avec les mots-clés #BYVAR1 (label ou nom de la 1re variable de l'instruction BY), #BYVAL1 (valeur courante de la première variable de l'instruction BY), et également #BYVAR2 et #BYVAL2, etc., autant que l'instruction BY compte de variables.

Exemple 1.13 – Gestion des blocs d'observations BY

```
ODS PDF FILE = "c:\temp\blocs BY.pdf" ;
OPTION NOBYLINE ;
ODS NOPROCTITLE ;
PROC SORT DATA = livre.voitures
           OUT = work.voitures ;
  BY type ;
```

```
RUN ;
TITLE1 "Cette sortie concerne l'ensemble des véhicules
        dont TYPE vaut #BYVAL1" ;
PROC FREQ DATA = work.voitures ;
  TABLE transmission ;
  BY type ;
RUN ;
TITLE ;
ODS PDF CLOSE ;
```



Cette sortie concerne l'ensemble des véhicules dont TYPE vaut Compacte

Mode de transmission				
transmission	Frequency	Percent	Cumulative Frequency	Cumulative Percent
4x4	1	6.25	1	6.25
propulsion	2	12.50	3	18.75
traction	13	81.25	16	100.00

Figure 1.13 — Résultat de l'exemple 1.13, le fichier BLOCS BY.PDF

1.4.3 La mise en page des sorties

Trois options principales régissent la pagination des sorties SAS : `ORIENTATION` (qui indique si le document est en portrait ou en paysage), `PAGESIZE` (la taille du papier sur lequel on imprimera la sortie) et `CENTER` (la justification des sorties).

L'option `ORIENTATION` peut (en version 9) voir sa valeur varier au cours de la création d'un même document RTF ou PDF. Il faut, pour signifier à l'ODS de prendre en compte la modification, ajouter au programme une instruction `ODS RTF ;` ou `ODS PDF ;` sans autre option.

L'option `PAPERSIZE` régit la largeur et la hauteur des sorties dans les destinations comme PDF et RTF : en particulier, elle est en relation avec la définition des marges dans le style global appliqué au document. Par défaut, la valeur de cette option est `LETTER` (soit 8,5 pouces de large pour 11 pouces de haut). Le format lettre américain est moins long et plus large qu'un A4 usuel (279×216 mm contre 297×210). Il est donc conseillé de modifier cette option. Il est à noter que dans SAS sous Windows, cette option est mise à jour par les choix enregistrés dans le menu FILE > PRINT SETUP (FICHIER > MISE EN PAGE pour les versions en français). Tandis que l'action d'un programme modifiant l'option `PAPERSIZE` est temporaire (jusqu'à la fin de la session SAS), la modification par la fenêtre de dialogue est permanente et rattachée à un ordinateur.

Enfin, l'option CENTER est activée par défaut ; si on la désactive, les sorties, les titres et les pieds de page sont justifiés à gauche.

```
OPTION CENTER | NOCENTER ;
OPTION ORIENTATION = PORTRAIT | LANDSCAPE ;
OPTION PAPERSIZE = LETTER | A4 | A3 ;
```

Tableau 1.2 — Destinations ODS et options pour la mise en page

Destination ODS	Centrage	Dimensions de la page	Portrait/paysage
LISTING	CENTER	PS, LS	ORIENTATION
HTML	non	non	non
OUTPUT	non	non	non
RTF	CENTER	PAPERSIZE	ORIENTATION
PDF	CENTER	PAPERSIZE	ORIENTATION
TAGSETS.EXCELXP	non	PAPERSIZE	option de l'instruction ODS

Les dimensions de pages dans la destination LISTING sont régies par des options différentes : PAGESIZE ou PS (nombre de lignes) et LINESIZE ou LS (nombre de caractères dans une ligne) adaptées aux éditions dans la police SAS Monospace à taille fixe. L'effet de l'option ORIENTATION sur la destination LISTING est de permuter temporairement les valeurs de LINESIZE et PAGESIZE.

1.4.4 La personnalisation d'une table des matières

Les destinations ODS qui créent des tables des matières (HTML avec la page CONTENTS, RTF et PDF avec l'option CONTENTS, PDF avec les signets ajoutés automatiquement et affichés sur la gauche du document) prennent des titres prédéfinis pour alimenter les différents niveaux de ces sommaires. Il est possible de les personnaliser pour rendre la navigation plus aisée aux lecteurs.

Attention cependant, un contrôle total des différents niveaux de hiérarchie n'est possible qu'avec les procédures Print, Freq, Tabulate et Report pour les tableaux, et avec les procédures graphiques comme Gchart et Gplot. Pour les autres procédures, les modifications sont envisageables à travers soit des options similaires à celles détaillées ci-dessous, soit une action sur leur *modèle tabulaire* (cf. le chapitre 5).

Les tableaux

Dans les procédures PRINT, TABULATE et REPORT (détaillées au chapitre 3), une option CONTENTS permet d'indiquer quel texte est à afficher pour chapeauter l'ensemble des sorties de la procédure. Dans le cas particulier de TABULATE qui peut produire plusieurs tableaux, on peut ajouter également cette option CONTENTS à

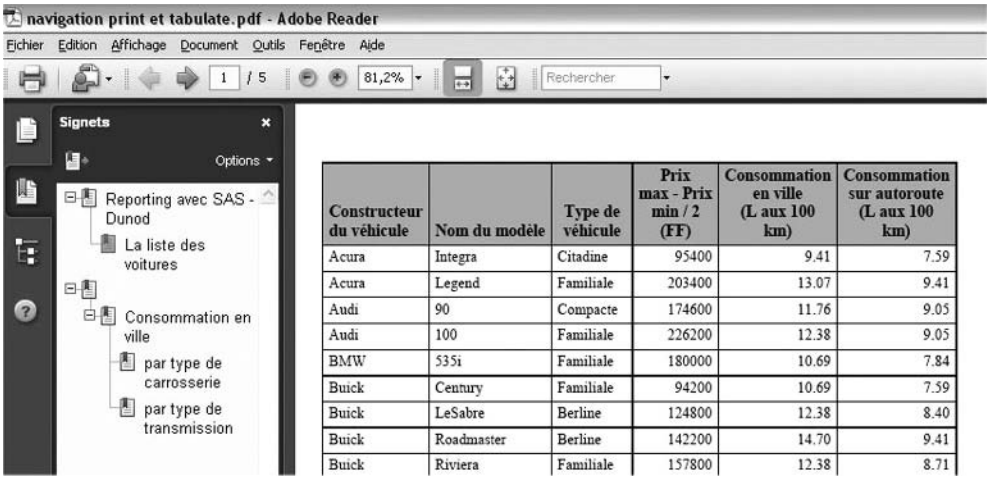
la fin de chaque instruction TABLE pour libeller les entrées de sommaire pour chacun des tableaux.

Quant au premier niveau de chaque sortie, correspondant à la procédure employée, on l'ajuste avec l'instruction ODS PROCLABEL, *réinitialisée à la fin de chaque procédure*.

```
ODS destination FILE="…" … ;
ODS PROCLABEL = "Pour PRINT : Titre 1" ;
PROC PRINT DATA = tableSAS CONTENTS = "Pour PRINT : Titre 2" … ;
   …
RUN ;
ODS PROCLABEL = "Pour REPORT : Titre 1" ;
PROC REPORT DATA = tableSAS CONTENTS = "Pour REPORT : Titre 2" … ;
   …
RUN ;
ODS PROCLABEL = "Pour TABULATE : Titre 1" ;
PROC TABULATE DATA = tableSAS CONTENTS = "Pour TABULATE : Titre 2" … ;
   …
   TABLE … / CONTENTS = "Pour TABULATE : Titre 3" … ;
RUN ;
ODS destination CLOSE ;
```

Exemple 1.14 – Gestion des libellés de sommaire avec les procédures Print et Tabulate

```
ODS PDF FILE = "c:\temp\navigation print et tabulate.pdf" ;
ODS PROCLABEL = "Reporting avec SAS - Dunod" ;
TITLE ;
PROC PRINT DATA = livre.voitures
       CONTENTS = "La liste des voitures"
         NOOBS LABEL ;
       VAR constructeur modele type prix_med conso_ville conso_auto ;
RUN ;
ODS PROCLABEL = "Consommation en ville" ;
PROC TABULATE DATA = livre.voitures CONTENTS = " " ;
       CLASS type transmission ;
       VAR conso_ville ;
       TABLE type="",
                conso_ville="Conso moyenne"*MEAN=""*F=NUMX7.1 /
                     BOX="Type de carrosserie"
                     CONTENTS = "par type de carrosserie" ;
       TABLE transmission="",
                conso_ville="Conso moyenne"*MEAN=""*F=NUMX7.1 /
                     BOX="Type de transmission"
                     CONTENTS = "par type de transmission" ;
RUN ;
ODS PDF CLOSE ;
```



Constructeur du véhicule	Nom du modèle	Type de véhicule	Prix max - Prix min / 2 (FF)	Consommation en ville (L aux 100 km)	Consommation sur autoroute (L aux 100 km)
Acura	Integra	Citadine	95400	9.41	7.59
Acura	Legend	Familiale	203400	13.07	9.41
Audi	90	Compacte	174600	11.76	9.05
Audi	100	Familiale	226200	12.38	9.05
BMW	535i	Familiale	180000	10.69	7.84
Buick	Century	Familiale	94200	10.69	7.59
Buick	LeSabre	Berline	124800	12.38	8.40
Buick	Roadmaster	Berline	142200	14.70	9.41
Buick	Riviera	Familiale	157800	12.38	8.71

Figure 1.14 — Résultat de l'exemple 1.14, le fichier NAVIGATION PRINT ET TABULATE.PDF

Les graphiques

Pour les procédures graphiques GPLOT et GCHART (cf. sections 4.4 et 4.5), l'option se nomme `DESCRIPTION` et elle s'ajoute aux instructions générant spécifiquement un graphique : `PLOT` et `BUBBLE` pour GPLOT, `VBAR`, `VBAR3D`, `HBAR`, `HBAR3D`, `PIE` et `PIE3D` dans GCHART. Il s'agit du libellé accolé au niveau du graphique lui-même. Pour le niveau chapeautant l'ensemble des graphiques générés par la procédure, c'est également l'instruction `ODS PROCLABEL` qui agit, comme précédemment (voir le résultat sur la figure 1.15).

Exemple 1.15 – Gestion des libellés de sommaire avec la procédure GPLOT

```
ODS PDF FILE = "c:\temp\navigation graphiques.pdf" ;
ODS PROCLABEL = "Graphiques exploratoires" ;
PROC GPLOT DATA = livre.voitures ;
       PLOT conso_ville * poids /
             DESCRIPTION = "Relation consommation vs poids" ;
       PLOT conso_ville * prix_med /
             DESCRIPTION = "Relation consommation vs prix" ;
RUN ; QUIT ;
ODS PDF CLOSE ;
```

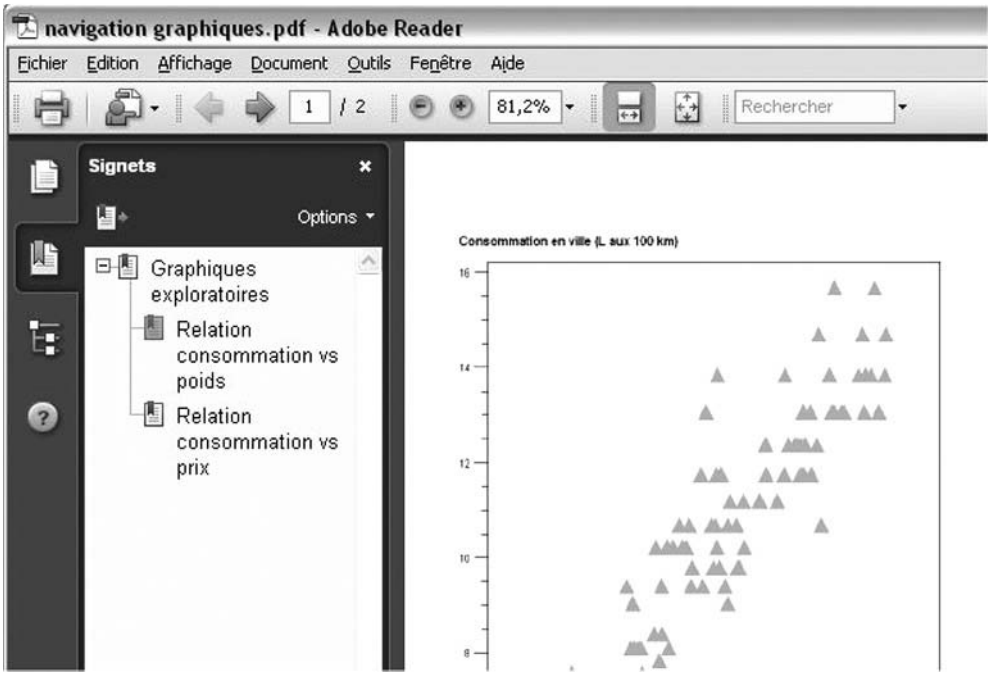


Figure 1.15 — Résultat de l'exemple 1.15, le fichier NAVIGATION GRAPHIQUES.PDF

2

La personnalisation avancée des sorties

Objectifs

Au moyen d'outils simples – formats, titres et pieds de page –, on peut insérer de la mise en forme précise dans l'ensemble des procédures SAS. Ces outils sont universels et fonctionnent particulièrement bien avec l'ODS.

On suppose dans tout ce chapitre une connaissance de la syntaxe de base de la procédure FORMAT (instruction `VALUE` pour la création de formats simples).

2.1 CRÉER DES FORMATS COMPLEXES

Les formats sont un outil extrêmement puissant pour le recodage de variables ou pour soigner la présentation d'un résultat. Nous nous penchons ici sur trois types particuliers de formats : les formats recouvrants (ou *multilabels*), les formats imbriqués qui « surchargent » la définition d'un format existant, et les formats calculés (appelés *pictures*) qui permettent de personnaliser l'affichage des nombres et des dates.

2.1.1 Les formats multilabels

Ces formats ne sont utilisables qu'avec deux procédures : MEANS et TABULATE. Ils permettent d'associer une valeur stockée dans la table SAS à plusieurs libellés

(donc plusieurs groupes) à l'affichage. On peut ainsi créer des tranches d'âges, par exemple, qui ne sont pas forcément disjointes. On peut également générer des récapitulatifs (sous-totaux). Pour cela, on rédige une instruction `VALUE` comme à l'habitude dans la procédure FORMAT, en prenant seulement soin de préciser l'option `MULTILABEL` à la suite du nom du format (qu'il soit numérique ou caractère). On définit ensuite les associations entre valeurs et libellés, avec la possibilité d'assigner plusieurs libellés à une même valeur.

```
PROC FORMAT ;
   VALUE nomFormat (MULTILABEL)
       … /* definition du format */
   ;
RUN ;
```

On utilise ensuite ces formats dans les procédures MEANS et TABULATE en précisant, dans l'instruction `CLASS` concernée, l'option `MLF` (*Multi-Label Format*). Les calculs tiennent alors compte des redondances dans les catégories.

Exemple 2.01 – Création et utilisation de formats multilabels

```
PROC FORMAT ;
       VALUE $marques (MULTILABEL)
           "Acura", "Honda", "Hyundai", "Infiniti", "Lexus", "Mazda",
           "Mitsubishi", "Nissan", "Subaru", "Suzuki", "Toyota"
               = "Japonaises"
           "Audi", "BMW", "Mercedes-Benz", "Saab", "Volkswagen", "Volvo"
               = "Européennes"
           "Buick", "Cadillac", "Chevrolet", "Chrylser", "Chrysler",
           "Dodge", "Eagle", "Ford", "Geo", "Lincoln", "Mercury",
           "Oldsmobile", "Plymouth", "Pontiac", "Saturn"
               = "Américaines"
           "Acura", "Honda", "Hyundai", "Infiniti", "Lexus", "Mazda",
           "Mitsubishi", "Nissan", "Subaru", "Suzuki", "Toyota", "Audi",
           "BMW", "Mercedes-Benz", "Saab", "Volkswagen", "Volvo"
               = "--- Ensemble hors USA"
       ;
       VALUE puissance (MULTILABEL)
                 0-<100 = "< 100 CV"
               100-<150 = "100-150 CV"
               150-<200 = "150-200 CV"
               100-<200 = "100-200 CV"
               200-HIGH = "> 200 CV"
               LOW-<200 = "< 200 CV"
       ;
RUN ;
ODS HTML FILE = "c:\temp\multilabel.htm" STYLE = journal ;
PROC MEANS DATA = livre.voitures MEAN MAXDEC=2 ;
       VAR conso_ville ;
       CLASS constructeur puissance / MLF ;
       FORMAT constructeur $marques. puissance puissance. ;
       WAYS 1 ;
RUN ;
ODS HTML CLOSE ;
```

Dans les résultats présentés sur la figure 2.1, on peut vérifier aux comptages d'observations que les catégories se recouvrent comme attendu : la table VOITURES ne compte bel et bien que 93 observations !

Analysis Variable : conso_ville Consommation en ville (L aux 100 km)

Puissance réelle (CV vapeur)	N Obs	Mean
100-150 CV	34	10.77
100-200 CV	60	11.47
150-200 CV	26	12.38
< 100 CV	19	8.24
< 200 CV	79	10.69
> 200 CV	14	13.11

Analysis Variable : conso_ville Consommation en ville (L aux 100 km)

Constructeur du véhicule	N Obs	Mean
--- Ensemble hors USA	43	10.66
Américaines	50	11.39
Européennes	12	11.77
Japonaises	31	10.23

Figure 2.1 — Résultat de l'exemple 2.01, le fichier MULTILABEL.HTM

2.1.2 Les formats imbriqués

Il est également possible de surcharger un format existant, qu'il soit défini par SAS ou que ce soit l'un des vôtres. Surcharger une définition signifie que l'on précise et modifie le comportement du format pour quelques cas particuliers, mais qu'on se repose sur la définition précédente pour le reste.

Pour cela, on construit un format avec la procédure FORMAT et l'instruction `VALUE` ; mais à droite du signe =, plutôt que d'indiquer un libellé à afficher, on donne un nom de format entre crochets.

```
PROC FORMAT < LIB = bibliothèqueDeSauvegarde > ;
   VALUE nomFormat
         valeur(s) = "valeur affichée"
         valeur(s) = [nomFormatExistant.]
    ;
RUN ;
```

Attention cependant, il n'est pas conseillé de pratiquer cet « héritage » de formats sur plus d'un niveau, sous peine de dysfonctionnements.

L'exemple 2.02 s'appuie sur le format SAS pvalue., qui permet d'afficher les probabilités de tests statistiques comme des nombres usuels, saufs ceux inférieurs à 0,0001 qui seront affichés « < 0.0001 » sans plus de précision. On souhaite utiliser ce format pour toutes les probabilités inférieures à 0,15 et afficher « non significatif » au-delà de 0,15.

Exemple 2.02 – Création et utilisation d'un format imbriqué

```
ODS OUTPUT pearsonCorr = work.correlations ;
PROC CORR DATA = livre.voitures ;
  VAR prix_max ; WITH regime_puiss plein braquage ;
RUN ;
PROC FORMAT ;
  VALUE prob
    0.15 - HIGH = "Non significatif"
    OTHER = [pvalue12.5]
  ;
RUN ;
ODS PDF FILE = "c:\temp\formats imbriqués.pdf" ;
TITLE1 "Corrélations avec le prix maximum du modèle" ;
PROC PRINT DATA = work.correlations NOOBS LABEL ;
  FORMAT pprix_max prob. ;
  LABEL prix_max = "Corrélation" pprix_max = "Prob > |r|" ;
RUN ;
TITLE ;
ODS PDF CLOSE ;
```

Corrélations avec le prix maximum du modèle

Variable	Label	Corrélation	Prob > |r|
regime_puiss	Régime de puissance maxi (tr/min)	0.02501	Non significatif
plein	Volume du réservoir (L)	0.58052	<.00001
braquage	Rayon de braquage (m)	0.38258	0.00015

Figure 2.2 — Résultat de l'exemple 2.02, le fichier FORMATS IMBRIQUES.PDF

2.1.3 Les formats calculés ou pictures

Les *pictures* sont des formats pour les variables numériques (y compris les dates), où on peut intégrer un calcul à la valeur affichée. Pratique pour présenter les valeurs numériques, le *picture* autorise toutes les fantaisies, appliqué à une date.

Le cas des nombres

Au contraire d'une instruction VALUE, la valeur indiquée entre guillemets pour l'affichage dans PICTURE n'est pas prise au pied de la lettre par SAS. Dans ce masque d'affichage, un 9 correspond à un chiffre qui sera systématiquement affiché et un 0 correspond à un chiffre qui ne sera affiché que s'il est non nul.

Si on veut inclure un texte avant la valeur, comme un signe moins, il faut le déclarer dans l'option PREFIX.

```
PROC FORMAT < LIB = bibliothèqueDeSauvegarde > ;
    PICTURE nomFormat < (ROUND) >
         plageValeurs = "valeur affichée" <(< PREFIX="…" > < MULT=nombre >)>
         plageValeurs = "valeur affichée" <(< PREFIX="…" > < MULT=nombre >)>
    ;
RUN ;
```

Les plages de valeurs sont à indiquer comme dans un format normal : les mots-clés HIGH, LOW et OTHER sont autorisés, ainsi que les < de part et d'autre du tiret (-) pour indiquer une exclusion d'une borne de l'intervalle considéré.

Les choses se compliquent si on souhaite afficher un nombre avec des décimales. L'option DECSEP, non détaillée ici, permet théoriquement d'indiquer à SAS quel caractère sert de séparateur décimal, mais cette option n'est pas tout le temps prise en compte (en présence de l'option MULT, elle est ignorée). Voici donc le raisonnement dans le cas général.

On doit jouer sur deux options : ROUND (à préciser entre parenthèses à côté du nom du format) pour que le résultat de la multiplication demandée soit arrondi à l'affichage comme dans un format classique (par défaut, le résultat sera tronqué !) ; et MULT à côté de chaque valeur à afficher afin d'indiquer quel multiplicateur sera appliqué au nombre tel qu'il est stocké.

Dans le masque indiqué entre guillemets, le dernier chiffre (0 ou 9) sera considéré comme le chiffre des unités dans le nombre à afficher. Du point de vue de SAS, la virgule décimale se trouve donc après le dernier chiffre du masque. On doit donc utiliser l'option MULT pour décaler la virgule en conséquence avant affichage. Dans les exemples suivants du tableau 2.1, à l'avant-dernière colonne, *ce chiffre est souligné*.

Tableau 2.1 — Exemples d'utilisation de l'option MULT en fonction du nombre de décimales à afficher

Valeur à afficher	Dans l'instruction PICTURE	Valeur * mult (avant arrondi)	Valeur *affichée* (après arrondi)
1234.567	"000 009"	1234.567	1 235
1234.567	"000 009" (MULT=10)	12345.67	12 346
1234.567	"000 009" (MULT=0.1)	123.4567	123

Valeur à afficher	Dans l'instruction PICTURE	Valeur * mult (avant arrondi)	Valeur *affichée* (après arrondi)
1234.567	"000 009,9" (MULT=0.1)	123.4567	12,3
1234.567	"000 009,99" (MULT=10)	12345.67	123,45
1234.567	"000 009,99" (MULT=100)	123456.7	1 234,57

Exemple 2.03 – Création et utilisation de formats *picture* sur des nombres

```
PROC FORMAT ;
  PICTURE milliers (ROUND)
    LOW -< 0 = "000 000 009,99" (MULT=100 PREFIX="-")
    0 - HIGH = "000 000 009,99" (MULT=100)
  ;
  PICTURE kEuros (ROUND)
    0 - HIGH = "000 009,99 k " (MULT=%SYSEVALF(0.1/6.55957))
  ;
RUN ;
PROC SQL OUTOBS=10 ;
  SELECT constructeur,
         modele,
         prix_max LABEL="Prix le + élevé en FF" FORMAT=milliers.,
         prix_max LABEL="Prix le + élevé" FORMAT=kEuros.
  FROM livre.voitures
  ;
QUIT ;
```

Constructeur du véhicule	Nom du modèle	Prix le + élevé en FF	Prix le + élevé
Acura	Integra	112 800,00	17,20 k€
Acura	Legend	232 200,00	35,40 k€
Audi	90	193 800,00	29,54 k€
Audi	100	267 600,00	40,80 k€
BMW	535i	217 200,00	33,11 k€
Buick	Century	103 800,00	15,82 k€
Buick	LeSabre	130 200,00	19,85 k€
Buick	Roadmaster	149 400,00	22,78 k€
Buick	Riviera	157 800,00	24,06 k€
Cadillac	DeVille	217 800,00	33,20 k€

Figure 2.3 — Résultat de l'exemple 2.03 au format PDF

Le cas des dates

Pour les dates, l'écriture d'un picture est plus riche, donc plus complexe. Elle nécessite d'abord d'indiquer le type de données formatées (`DATE`, `DATETIME`, `TIME`) dans une option `DATATYPE` située après chaque série de valeurs affichées. Il est conseillé d'indiquer la longueur de la plus longue valeur affichée dans une option `DEFAULT` située, elle, après le nom du format.

```
PROC FORMAT < LIB = bibliothèqueDeSauvegarde > ;
    PICTURE nomFormat (DEFAULT = longueurParDéfaut)
         plageValeurs = "valeur affichée"
                                  (DATATYPE = DATE|TIME|DATETIME)
    ;
RUN ;
```

Les valeurs affichées peuvent comprendre les expressions suivantes :

- pour une date ou un datetime :
  - `%a` = nom du jour de la semaine en abrégé ;
  - `%A` = nom du jour de la semaine en intégralité ;
  - `%b` = nom du mois en abrégé ;
  - `%B` = nom du mois en intégralité ;
  - `%d` = jour du mois (de 1 à 31) sans zéro préalable ;
  - `%0d` = jour du mois sur deux chiffres (y compris zéro préalable) ;
  - `%j` = jour dans l'année (de 1 à 366) sans zéro préalable ;
  - `%0j` = jour dans l'année sur 3 chiffres (y compris zéro préalable) ;
  - `%m` = numéro du mois (de 1 à 12) sans zéro préalable ;
  - `%0m` = numéro du mois sur 2 chiffres (y compris zéro préalable) ;
  - `%U` = numéro de semaine (de 0 à 53) avec le dimanche comme 1er jour de la semaine ;
  - `%w` = numéro du jour de la semaine (de 1=dimanche à 7=samedi) ;
  - `%y` = année sur 1 ou 2 chiffres (de 0 à 99) ;
  - `%0y` = année sur 2 chiffres (de 00 à 99) ;
  - `%Y` = année sur 4 chiffres.
- Pour un temps ou un datetime :
  - `%H` = heure (de 0 à 23) sans zéro préalable ;
  - `%0H` = heure (de 00 à 23) ;
  - `%I` et `%0I` = heure par cycles de 12 heures (de 0 ou 00 à 12) ;
  - `%M` et `%0M` = minutes ;
  - `%p` = matin ou après-midi ;
  - `%S` et `%0S` = secondes.

Les noms (de mois, de jour) sont en français si l'option `LOCALE` vaut `FRENCH` et sinon en anglais. Cette option est indépendante de la langue de la session SAS.

Exemple 2.04 – Création et utilisation d'un format *picture* sur des dates

```
PROC FORMAT ;
  PICTURE mois (DEFAULT = 40)
```

```
    LOW-HIGH = "Mois de %B %Y" (DATATYPE=DATE)
  ;
RUN ;
PROC TABULATE DATA = sashelp.prdsale F = milliers. ;
       CLASS month ;
       FORMAT month mois. ;
       VAR actual ;
       TABLE month = "", actual = "Ventes" * SUM = "" ;
RUN ;
```

	Ventes
Mois de janvier 1993	29 813,00
Mois de février 1993	29 584,00
Mois de mars 1993	29 873,00
Mois de avril 1993	30 581,00
Mois de mai 1993	31 617,00
Mois de juin 1993	33 605,00
Mois de juillet 1993	33 578,00
Mois de août 1993	31 160,00

Figure 2.4 — Résultat de l'exemple 2.04 au format PDF

2.2 LES INSTRUCTIONS DE TITRES ET DE PIEDS DE PAGE

Les instructions `TITLE` et `FOOTNOTE`, déclinées sur dix niveaux, sont assez simples dans leur utilisation basique : on indique le texte à y faire figurer. Dans cette partie, on présente la mise en forme de ces textes. Uniquement visibles à travers les destinations ODS compatibles (toutes sauf LISTING et OUTPUT) des options permettent de contrôler la mise en forme des textes intégrés aux titres.

```
TITLE1 <BOLD> <ITALIC> <JUSTIFY=LEFT|CENTER|RIGHT>
       <FONT="nom police"> <HEIGHT=xPT>
       <COLOR=couleurPolice> <BCOLOR=couleurFond> "texte" ;
```

Cette syntaxe s'applique évidemment à tous les niveaux de titre, de `TITLE1` à `TITLE10`, et vaut aussi pour les pieds de page, instructions `FOOTNOTE1` à `FOOTNOTE10`. Il est également possible de scinder le texte du titre en plusieurs morceaux pour les faire bénéficier d'une mise en forme distincte, au sein d'un même niveau de titre :

```
TITLE1 <options> "texte 1" <options> "texte 2" … ;
```

Les différentes options permettent :

- pour `BOLD` de mettre le texte en gras ;
- pour `ITALIC` de le mettre en italique ;
- pour `JUSTIFY` de gérer l'alignement des caractères (`LEFT` = aligné à gauche, `RIGHT` aligné à droite, `CENTER`, centré) ;
- pour `FONT` de gérer la police de caractères utilisée ;
- pour `HEIGHT` de gérer la taille des caractères (`12PT` correspondant à une police de taille 12 dans un logiciel de traitement de texte) ;
- pour `COLOR` et `BCOLOR`, de gérer respectivement la couleur de la police et la couleur de fond du paragraphe de titre correspondant. La liste des codes couleurs licites est la suivante :
  - Les mots-clés correspondant à des couleurs prédéfinies : `BLACK`, `BLUE`, `BROWN`, `CHARCOAL`, `CREAM`, `CYAN`, `GOLD`, `GRAY`, `GREEN`, `LILAC`, `LIME`, `MAGENTA`, `MAROON`, `OLIVE`, `ORANGE`, `PINK`, `PURPLE`, `RED`, `ROSE`, `SALMON`, `STEEL`, `TAN`, `VIOLET`, `WHITE`, `YELLOW`. D'autres mots-clés sont également définis selon les sessions SAS et la plate-forme (Windows, Unix, MVS). On peut les lister avec la proc Registry[1].
  - Les codes RGB (habituellement utilisés dans les feuilles de styles des pages HTML) où les six « chiffres » du code hexadécimal sont à faire précéder de `CX` (en lieu et place du # qu'on trouve en HTML). On les énonce entre guillemets : l'option `COLOR="CX0000AA"` écrit le titre en bleu foncé. On trouve aisément sur Internet des nuanciers indiquant les codes RGB associés aux différentes couleurs.
  - Les niveaux de gris (`GRAY`*hh*) : les *hh* sont à remplacer par deux chiffres en code hexadécimal, de `00` (noir) à `FF` (blanc).

Exemple 2.05 – Personnalisation des titres d'un document

```
ODS PDF FILE = "c:\temp\titres.pdf" ;
TITLE1 HEIGHT=24PT COLOR=GRAY60 "Voitures disponibles" ;
TITLE2 HEIGHT=18PT COLOR=GRAY60 "sur le marché américain" ;
TITLE4 J=L "Echantillon" J=R "Données de 1993" ;
PROC PRINT DATA = livre.voitures LABEL NOOBS ;
       VAR constructeur modele type puissance cylindree ;
RUN ;
TITLE ;
ODS PDF CLOSE ;
```

1. On utiliser la code suivant : `PROC REGISTRY LIST STARTAT="colornames\html" ; RUN ;` pour lister dans la fenêtre Log les codes couleurs disponibles sur la session SAS.

Voitures disponibles
sur le marché américain

Echantillon *Données de 1993*

Constructeur du véhicule	Nom du modèle	Type de véhicule	Puissance réelle (CV vapeur)	Cylindrée (L)
Acura	Integra	Citadine	140	1.8
Acura	Legend	Familiale	200	3.2
Audi	90	Compacte	172	2.8
Audi	100	Familiale	172	2.8
BMW	535i	Familiale	208	3.5

Figure 2.5 — Résultat de l'exemple 2.05, le fichier TITRES.PDF

2.3 L'INSERTION DE MISE EN FORME DANS UNE SORTIE

On peut souhaiter insérer du langage RTF ou HTML pour améliorer la forme des textes ou intégrer de manière détournée des éléments spécifiques à un document. Pour certains éléments, il existe des codes prédéfinis (mise en forme intégrée ou *in-line formatting*). L'insertion de caractères spéciaux sera également présentée en fin de section.

2.3.1 L'insertion de code HTML ou RTF dans les sorties SAS

On peut également, pour augmenter les possibilités de mise en forme, s'appuyer sur le langage de mise en forme dans lequel le document généré est rédigé : RTF ou HTML. Les autres langages (XML pour TAGSETS.EXCELXP, PostScript pour ODS PDF en particulier), trop complexes, ne seront pas abordés ici.

Attention : on n'a pas de garantie *a priori* que le logiciel affichant le document comprendra correctement ce qu'on lui demande. Les différents navigateurs Internet ne rendent pas le HTML de manière identique ; et si on ouvre la page HTML dans Excel ou Word, on aura encore d'autres surprises !

Le code HTML ou RTF peut être intégré dans des titres, des pieds de page, et également dans les labels des variables ou même dans leurs valeurs (on utilisera alors un format).

Le code HTML

Il s'agit d'un langage de balises dont la syntaxe est désormais assez répandue. On trouve sans problème de nombreux sites Internet qui décrivent l'ensemble des balises utilisables, avec leurs paramètres et leurs fonctions.

La balise <IMG> permet d'insérer une image. Son option SRC précise quelle image (on utilisera des chemins relatifs, en stockant les images dans un répertoire proche de celui du fichier HTML produit). Dans l'exemple 2.06, on utilise aussi la balise <MARQUEE> pour avoir un titre qui défile de gauche à droite de l'écran.

Si on souhaite insérer du code HTML dans une valeur ou un label de variable, il faut préciser que les caractères < et > ne doivent pas être pris pour des caractères « affichables » (ils sont protégés par défaut, pour ne pas être confondus avec des balises, justement). Ce n'est possible que dans les procédures Print, Tabulate et Report (plus de détails sur la syntaxe des options STYLE dans le chapitre 3 consacré à ces procédures) avec l'option PROTECTSPECIALCHARS=OFF.

Exemple 2.06 – Insertion de code HTML dans une sortie SAS

```
PROC FORMAT ;
  VALUE $trans
    "propulsion" = "<IMG SRC='prop.jpg'>     propulsion"
    "traction"   = "<IMG SRC='traction.jpg'> traction"
    "4x4"        = "<IMG SRC='4x4.jpg'>      4 x 4" ;
RUN ;
ODS HTML FILE = "c:\temp\insertion HTML.htm" STYLE = JOURNAL ;
TITLE1 "<MARQUEE DIRECTION=RIGHT WIDTH='30%'>Statistiques sur la consommation</
MARQUEE>" ;
PROC TABULATE DATA = livre.voitures F=NUMX12.2 ;
  VAR conso_: ;
  CLASS transmission ;
  CLASSLEV transmission / STYLE = [PROTECTSPECIALCHARS=OFF] ;
  FORMAT transmission $trans. ;
  TABLE transmission="",
       (conso_ville conso_auto)*MEAN="" ;
RUN ;
TITLE ;
ODS HTML CLOSE ;
```

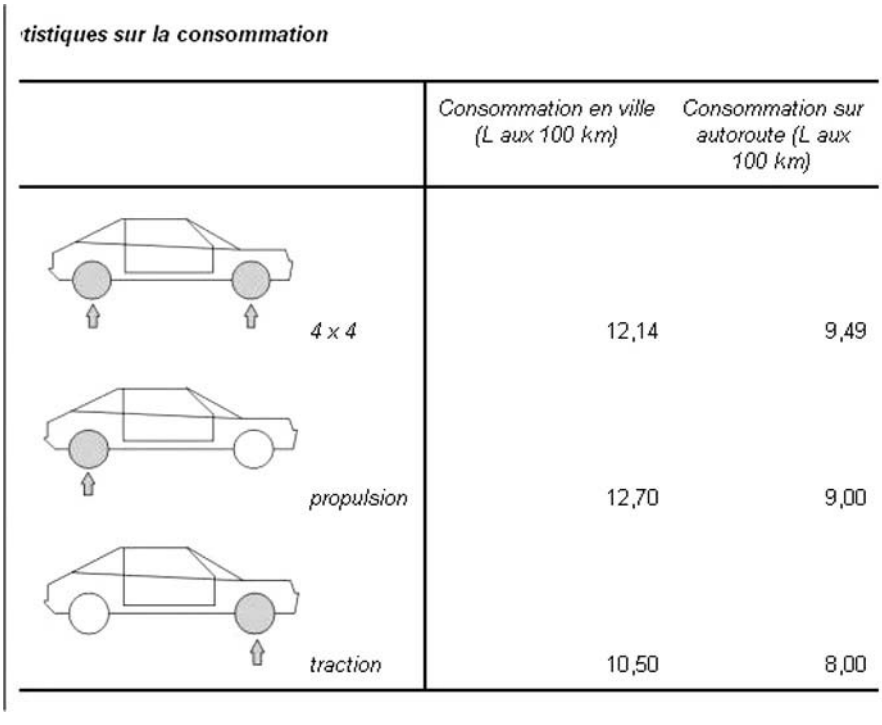


Figure 2.6 — Résultat de l'exemple 2.06, le fichier INSERTION HTML.HTM

Le code RTF

Le langage RTF, bien que compris par tous les logiciels de traitement de texte, est largement moins documenté et vulgarisé que le HTML. On trouve cependant sur Internet un document volumineux (et en anglais), *Rich Text Format (RTF) specification version 1.5*, exhaustif sur ce qu'on peut écrire comme code dans un document de ce format.

Le tableau 2.2 reprend les principales syntaxes pour la mise en forme d'un texte, ainsi que diverses instructions utiles comme les sauts de page.

À noter que pour annuler ces effets, on reprend le même code et on le termine par un zéro : par exemple `\b en gras \b0 pas en gras` et `\animtext1 texte animé \animtext0 texte pas animé`.

Tableau 2.2 — Principaux éléments de langage RTF (respecter la casse !) pour la mise en forme

Commande	Usage
\plain	Ramène au style par défaut
\animtext	Texte avec animation : \animtext1 = néon ; \animtext2 = fond clignotant ; \animtext3 = feux d'artifice ; \animtext4 = encadré pointillé animé (fourmis) noir ; \animtext5 = le même en rouge ; \animtext6 = texte « vibrant »
\b, \i	Texte en gras, en italique
\embo, \impr	Texte en relief, « incrusté » dans la feuille (relief inverse)
\sub, \super	Texte en indice, en exposant ; \nosupersub annule les mises en indice et en exposant
\fsTAILLE	Change la taille du texte (TAILLE est à remplacer par un nombre en demi-points, c'est-à-dire le double de la taille indiquée dans Word).
\outl, \shad	Contour, ombré
\strike	Barré
\ul	Souligné (ligne pleine) ; variantes : \uld pour le soulignement pointillé, \uldb pour le soulignement double ; \ulnone pour annuler les soulignements
\tabTAILLE	Tabulation (dont la taille est donnée en *twips* : le *twip* vaut 1/12 de point, ce qui fait que 1 cm représente environ 567 *twips*)
\liTAILLE	Indentation d'un paragraphe (dont la taille est donnée en *twips* : le *twip* vaut 1/12 de point, ce qui fait que 1 cm représente environ 567 *twips*)
{\page}	Saut de page
\scaps	Texte en petites majuscules

Le plus simple pour l'incrustation de code RTF est de la signaler expressément à SAS. Pour cela, on les inclut entre guillemets, précédées d'un `R` lui-même précédé d'un caractère d'échappement.

Ce caractère, dont le choix est libre (en prendre un qui a peu de chances de se retrouver dans les sorties : accent circonflexe, astérisque, dièse sont de bons candidats), sera indiqué dans une instruction `ODS ESCAPECHAR`. Ce choix est gardé en mémoire jusqu'à la fin de la session SAS, ou jusqu'à exécution d'une instruction identique. Dans la suite de la syntaxe, on choisit l'accent circonflexe comme caractère d'échappement.

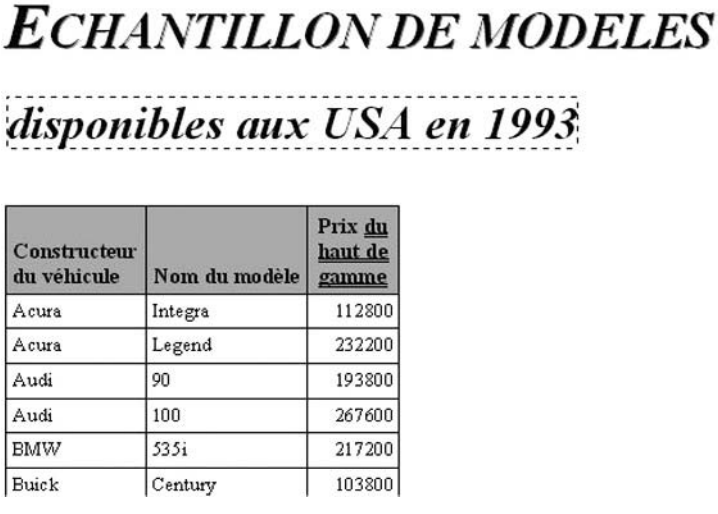
```
ODS ESCAPECHAR = "^" ;
```

Pour le RTF, on écrira donc `^R"mon code RTF"` dans les titres, labels et formats. Afin de ne pas être dépendant d'un choix de caractère d'échappement, on peut éga-

lement le remplacer par le texte (*ESC*). On peut donc également écrire (*ESC*)R"mon code RTF" pour un effet identique.

Exemple 2.07 – Insertion de code RTF dans une sortie SAS

```
ODS ESCAPECHAR="^" ;
ODS RTF FILE = "c:\temp\code RTF.doc" BODYTITLE ;
TITLE1 HEIGHT = 32pt '^R"{\embo \scaps Echantillon de modèles}"' ;
TITLE3 HEIGHT = 24pt '^R"{\animtext4 disponibles aux USA en 1993}"' ;
PROC PRINT DATA = livre.voitures LABEL NOOBS ;
       VAR constructeur modele prix_max ;
       LABEL prix_max = "Prix ^R'{\uldb du haut de gamme}'" ;
RUN ;
TITLE ;
ODS RTF CLOSE ;
```



ECHANTILLON DE MODELES

disponibles aux USA en 1993

Constructeur du véhicule	Nom du modèle	Prix du haut de gamme
Acura	Integra	112800
Acura	Legend	232200
Audi	90	193800
Audi	100	267600
BMW	535i	217200
Buick	Century	103800

Figure 2.7 — Résultat de l'exemple 2.07, le fichier CODE RTF.DOC

2.3.2 L'insertion d'éléments prédéfinis

On peut utiliser dans les sorties SAS (aussi bien dans les titres, pieds de pages, que les textes édités par les procédures elles-mêmes) des mots-clés prédéfinis qui influencent la mise en forme : mise en exposant ou en indice, insertion de saut de page, du numéro de page, ou encore du caractère spécial †. Afin de permettre à SAS de repérer ces mots-clés et de les convertir en instructions idoines pour le format de document produit par l'ODS, on les fait précéder d'un caractère d'échappement.

Les mots-clés suivants (à orthographier impérativement **en minuscules** !) sont prédéfinis pour les destinations PDF et RTF (et pour certains, HTML) de l'ODS :

- ^n insère un retour à la ligne ;
- ^_ insère un espace insécable ;

- `^w` indique un point où il est possible de revenir à la ligne (sans obligation si la ligne tient dans la largeur de la page) ;
- `^m` marque un emplacement pour une indentation de la ligne suivante ;
- `^-2n` insère un retour à la ligne avec une indentation au niveau de la marque `^m` ;
- `^{thispage}` insère le numéro de page et `^{lastpage}` le numéro de la dernière page ;
- `^{pageof}` insère le numéro de page sous la forme « Page 1 of 3 » ;
- `^{dagger}` insère le caractère † ;
- `^{super monTexteEnExposant}` met un texte en exposant ;
- `^{sub monTexteEnIndice}` met un texte en indice.

La mise en exposant permet par exemple de créer un système de notes de bas de page entre un label de variable et un *footnote*, comme dans l'exemple 2.09 (sortie sur la figure 2.8).

Exemple 2.08 – insertion d'éléments spéciaux prédéfinis dans une sortie SAS

```
ODS ESCAPECHAR = "^" ;
ODS PDF FILE = "c:\temp\spécial.pdf" ;
TITLE1 "Véhicules disponibles sur le marché américain^{super 1}" ;
FOOTNOTE1 HEIGHT=8PT "^{super 1}échantillon recueilli en 1993" ;
FOOTNOTE2 HEIGHT=8PT "^{super 2}prix de l'entrée de gamme en FF de 1993" ;
PROC PRINT DATA = livre.voitures (OBS = 10) LABEL NOOBS ;
       VAR constructeur modele prix_min ;
       LABEL prix_min = "Prix^{super 2}" ;
RUN ;
TITLE ;
FOOTNOTE ;
ODS PDF CLOSE ;
```

Véhicules disponibles sur le marché américain[1]

Constructeur du véhicule	Nom du modèle	Prix[2]
Acura	Integra	77400
Acura	Legend	175200
Audi	90	155400
Audi	100	184800
BMW	535i	142200
Buick	Century	85200
Buick	LeSabre	119400
Buick	Roadmaster	135600

[1]échantillon recueilli en 1993
[2]prix de l'entrée de gamme en FF de 1993

Figure 2.8 — Résultat de l'exemple 2.08, le fichier SPECIAL.PDF

2.3.3 L'insertion d'instructions de mise en forme

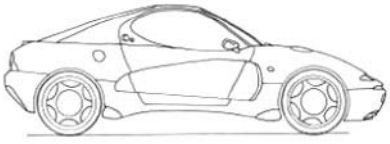
On peut également faire de la mise en forme de tout ou partie du texte présenté dans les sorties d'une procédure, ou dans le texte d'un titre ou d'une note de pied de page. Toujours en supposant que l'on a choisi l'accent circonflexe comme caractère d'échappement, on écrira devant la partie à mettre en forme `^S={…}` ; on réinitialisera la mise en forme par `^S={}`. Les éléments de mise en forme qui peuvent être inclus entre les accolades sont résumés dans le tableau 2.3.

Tableau 2.3 — Principaux éléments de style pour la mise en forme

Nom de l'élément	Valeurs possibles	Agit sur…
BACKGROUND	code couleur (cf. section 2.2.1)	Couleur de fond
FOREGROUND	code couleur (cf. section 2.2.1)	Couleur de la police
FLYOVER	texte	Info-bulle (HTML) ou note (PDF) ; pas supporté en RTF
URL	adresse de fichier (relative ou absolue)	Lien hypertexte
JUST	LEFT, RIGHT, CENTER	Justification horizontale
VJUST	TOP, BOTTOM, MIDDLE	Justification verticale
FONT_FACE	nom de police entre guillemets	Police utilisée (pour un document PDF, attention à avoir déclaré la police avec la proc FONTREG ; cf. 1.2.5).
FONT_SIZE	de 1 à 7, ou xPT avec x la taille en points	Taille de la police utilisée (pour le HTML, de 1 pour la plus petite à 7 pour la plus grande).
FONT_STYLE	ITALIC, ROMAN	Police italique ou droite
FONT_WEIGHT	BOLD, MEDIUM, LIGHT	Police grasse, normale ou fine
PREIMAGE ou POSTIMAGE	adresse d'un fichier image GIF ou JPEG (relative de préférence en HTML, absolue en RTF et PDF)	Insertion d'une image avant ou après le texte
ASIS	ON, OFF	Affichage des blancs situés à la gauche du texte (ASIS=ON) ; par défaut, ASIS=OFF

Exemple 2.09 – insertion de mise en forme dans une sortie SAS

```
ODS PDF FILE = "c:\temp\styles.pdf" ;
ODS ESCAPECHAR = "^" ;
TITLE1 "^S={PREIMAGE='c:\temp\voiture.jpg'}" ;
TITLE2 "^S={URL='http://www.dunod.com'}Notre site" ;
PROC PRINT DATA = livre.voitures NOOBS LABEL ;
  VAR constructeur modele puissance cylindree ;
  LABEL puissance = "^S={FLYOVER='en CV'}Puissance ^S={}du haut de gamme"
    cylindree = "^S={FLYOVER='en Litres'}Cylindrée ^S={}du haut de gamme" ;
RUN ;
TITLE ;
ODS PDF CLOSE ;
```



Notre site

Constructeur du véhicule	Nom du modèle	Puissance du haut de gamme	Cylindrée ... gamme (en Litres)
Acura	Integra	140	1.8
Acura	Legend	200	3.2
Audi	90	172	2.8

Figure 2.9 — Résultat de l'exemple 2.09, le fichier STYLES.PDF

2.3.4 Les caractères spéciaux

Il est souvent utile de pouvoir intégrer aux sorties (dans les titres, mais aussi les labels et les valeurs) des caractères particuliers : lettres de l'alphabet grec pour les unités de mesure, symboles mathématiques, pictogrammes usuels… Autant de caractères pas forcément possibles à taper au clavier : on peut soit les retrouver en les affichant via une police spéciale, soit chercher le numéro ASCII que porte le caractère dans toutes les polices, et utiliser ce numéro dans les sorties SAS (pratique pour les caractères spécifiques à certaines langues, comme les « o barrés » scandinaves).

L'utilisation de polices comme Symbol et Wingdings

Il est possible d'installer sur un ordinateur un grand nombre de polices, dont certaines rendent des caractères spéciaux : caractères de l'alphabet grec et notations mathématiques (SYMBOL), pictogrammes (WINGDINGS et dérivés, ainsi que WEBDINGS), alphabets arabes, hébreu, cyrillique, chinois, coréen, etc. Le tableau 2.4 indique à quoi correspondent les caractères usuels du clavier dans les polices SYMBOL pour les caractères grecs et mathématiques et WINGDINGS, qui sont les plus courantes.

Attention : si vous souhaitez employer des polices « spéciales » dans une destination ODS PDF, pensez à les enregistrer auprès de SAS *via* la procédure FONTREG (cf. section 1.2.5). Si vous les utilisez dans les autres destinations, ces caractères ne seront vus correctement que si les polices en question sont installées sur l'ordinateur qui ouvre le document : n'abusez donc pas des polices trop exotiques !

Tableau 2.4 — Affichage des caractères courants en police Symbol et Wingdings

Police courante			Police Symbol			Police Wingdings		
a	b	c	α	β	χ	♋	♌	♍
d	e	f	δ	ε	ϕ	♎	♏	♐
g	h	i	γ	η	ι	♑	♒	♓
j	k	l	φ	κ	λ	🙰	&	●
m	n	o	μ	ν	ο	❍	■	□
p	q	r	π	θ	ρ	❒	❑	❒
s	t	u	σ	τ	υ	⬧	⧫	◆
v	w	x	ϖ	ω	ξ	❖	⬧	⊠
y	z	,	ψ	ζ	,	🖂	⌘	🖃
;	:	!	;	:	!	🖮	🖳	🖉
ù	*	$	⎤	*	∃	▫	🖂	🖋
&	é	«	&	⎡	∀	🕮	🡅	✂
‘	(	-	ɔ	(	–	🕯	☎	🖃
è	_	ç	⎣	_	⎢	🖹	🡆	♉
à	)	=	◊	)	=	🡒	🕭	🖬
~	#	{	~	#	{	”	✁	🕆
[	|	`	[	|		☯	✿	♊
\	^	@	∴	⊥	≅	🕉	♈	🖍

Police courante			Police Symbol			Police Wingdings		
A	B	C	Α	Β	Χ	🖒	🖓	🖔
D	E	F	Δ	Ε	Φ	🖑	🖗	🖙
G	H	I	Γ	Η	Ι	🖐	🖕	🖖
J	K	L	ϑ	Κ	Λ	☺	😐	☹
M	N	O	Μ	Ν	Ο	💣	☠	🕱
P	Q	R	Π	Θ	Ρ	🏳	✈	☼
S	T	U	Σ	Τ	Υ	💧	❄	✞
V	W	X	ς	Ω	Ξ	✞	✠	✠
Y	Z	?	Ψ	Ζ	?	✡	☪	🕴
.	/	§	.	/	♣	🖃	🖃	▪
%	μ	£	%	∝	≤	🕭	✪	⭘
1	2	3	1	2	3	🗁	🗎	🗏
4	5	6	4	5	6	🗐	🗄	⧖
7	8	9	7	8	9	🖭	🖰	🖲
0	°	+	0	°	+	🗀	🖧	🖳
²	<	>	″	<	>	✧	🖫	🕈
^	¨		⊥	♦		♈	□	
]	}	¤	]	}	/	☸	“	⊙

Dans l'exemple 2.10, pour insérer des caractères grecs (khi et pi) on utilise respectivement c et p en police SYMBOL dans les labels d'une variable, afin de présenter de manière plus esthétique les sorties d'un test du khi-2 (procédure Freq).

Exemple 2.10 – insertion de caractères spéciaux en utilisant la police Symbol

```
PROC FORMAT ;
  VALUE conso
     0 -<6    = "<6"
     6 -<8    = "6-8"
     8 -<10   = "8-10"
    10 - HIGH = ">10"
  ;
RUN ;
ODS EXCLUDE ALL ;
ODS OUTPUT chiSq = work.chi2 ;
PROC FREQ DATA = livre.voitures ;
  TABLE type * conso_auto / CHISQ ;
  FORMAT conso_auto conso. ;
RUN ;
```

```
ODS SELECT ALL ;
TITLE ;
OPTION NODATE NONUMBER ;
ODS ESCAPECHAR = "^" ;
ODS PDF FILE = "c:\temp\chi2.pdf" ;
PROC PRINT DATA = work.chi2 (OBS=1) LABEL NOOBS SPLIT="#" ;
  VAR value prob df ;
  LABEL value = "^S={FONT_FACE='Symbol'}c^S={}^{super 2}"
        prob = "^S={FONT_FACE='Symbol'}p^S={}-value#
                (H^{sub 0}=indépendance)"
          df = "Degrés#de#liberté" ;
  FORMAT value NUMX12.2 prob NUMX8.5 ;
RUN ;
ODS PDF CLOSE ;
```

χ^2	π-value (H_0=indépendance)	Degrés de liberté
107,90	0,00000	15

Figure 2.10 — Résultat de l'exemple 2.10, le fichier CHI2.PDF

L'insertion d'un caractère par son numéro : la fonction BYTE

Certains caractères spéciaux existent dans la plupart des polices, mais ne sont pas directement accessibles au clavier. Dans un logiciel de traitement de texte, on passerait par un menu d'insertion particulier. C'est le cas par exemple des signes de copyright, et de la plupart des symboles mathématiques.

On peut insérer ces caractères quand on connaît leur numéro dans la norme de codage ASCII ; par exemple, le signe copyright correspond au numéro 0169. Pour connaître ce numéro, SAS propose la fonction `RANK`. On écrira donc `RANK("©")` pour avoir en retour le nombre 169. On peut aussi utiliser le programme Windows appelé « Table des caractères » ou consulter sur Internet les codes ASCII du caractère souhaité. Pour utiliser ce code dans SAS, on utilise la syntaxe `%SYSFUNC(BYTE(numéro))`. La fonction `BYTE` a pour vocation de générer un caractère à partir de son code ASCII ; la macro-fonction `%SYSFUNC` permet d'utiliser `BYTE` dans n'importe quel contexte : titre, format, label.

Exemple 2.11 – Insertion de caractères spéciaux avec la fonction BYTE

```
PROC SQL ;
  CREATE TABLE work.stats AS
    SELECT type,
           MEAN(conso_auto) AS moyenne,
           MEAN(conso_auto)-2*STD(conso_auto) AS inf,
           MEAN(conso_auto)+2*STD(conso_auto) AS sup
    FROM livre.voitures  GROUP BY type  ;
QUIT ;
```

```
PROC TABULATE DATA = work.stats F = NUMX12.2 ;
  CLASS type ;
  VAR moyenne inf sup ;
  TABLE type="", moyenne="Conso autoroute moyenne"*SUM=""
        SUM="Moyenne %SYSFUNC(BYTE(0177)) 2 écarts-types"*(inf="" sup="") ;
RUN ;
```

	Conso autoroute moyenne	Moyenne ± 2 écarts-types	
Berline	8,82	7,97	9,67
Citadine	6,77	4,87	8,68
Compacte	7,94	6,43	9,45
Familiale	8,88	7,20	10,55
Monospace	10,79	9,35	12,23
Sportive	8,29	6,26	10,32

Figure 2.11 — Résultat de l'exemple 2.11 au format PDF

3

Les listes, les tableaux, les rapports

Objectifs

Ce chapitre traite de l'emploi de trois procédures : PRINT, TABULATE et REPORT. Si la première est d'un emploi courant – quoi de plus incontournable que de lister une table SAS ? –, les deux autres sont souvent redoutées voire esquivées par ceux qui programment en SAS. Certes, leur syntaxe et leur logique ne sont pas forcément triviales, mais toutes deux sont capables de merveilles de mise en forme, et des partenaires privilégiés de l'ODS. Seul l'objet `ODSOUT` de l'étape Data, expérimental en version 9.1 et présenté au chapitre 8, peut revendiquer autant de souplesse dans le formatage des résultats.

Pour chaque procédure, nous présenterons ses fonctionnalités de base puis nous entrerons dans le détail des mises en forme plus complexes et de leurs liens avec l'ODS – mise en forme conditionnelle, couleurs, liens...

3.1 LES LISTES AVEC LA PROCÉDURE PRINT

La procédure PRINT a vocation à lister le contenu d'une table SAS. Elle permet d'assortir ce listing de comptages et de totaux, ainsi que de ruptures selon certaines variables (listing éclaté en blocs) et de sous-totaux. Des options de style permettent d'agir sur l'aspect du listing, de manière figée ou dynamique en fonction des valeurs listées.

3.1.1 La syntaxe de base de la procédure PRINT

Les grands traits de la procédure PRINT étant largement décrits par ailleurs[1], ils ne seront que rapidement abordés ici.

```
PROC PRINT DATA = tableSAS < N <="intitulé(s)> > < NOOBS > < LABEL > ;
  VAR listeVariable(s) ;
  SUM variable(s)Totalisée(s) ;
  BY variable(s)Rupture(s) ;
    PAGEBY variableRupture ;
RUN ;
```

L'instruction PAGEBY est dépendante de la présence de l'instruction BY, et ne peut indiquer qu'une seule variable. À chaque changement de valeur de celle-ci, une nouvelle page sera démarrée dans le document produit (sauf s'il est au format HTML où la notion de page n'existe pas).

L'instruction BY génère à chaque rupture une ligne de texte automatisé indiquant quelle est la nouvelle combinaison de valeurs des variables du BY. Il est possible d'éliminer cette ligne avec l'option système NOBYLINE. On peut alors la remplacer par un titre dans lequel le mot-clé #BYVAL1 (et #BYVAL2, etc., si BY indique plusieurs variables) remplace la valeur de la 1re variable citée dans BY. Attention, les titres ne sont affichés (hors HTML) que lors d'un saut de page : il faut donc penser à l'instruction PAGEBY pour personnaliser ses titres à chaque BY.

Exemple 3.01 – Procédure PRINT, blocs BY et titres personnalisés

```
PROC SORT DATA = livre.voitures OUT = work.voitures ;
       BY constructeur prix_max ;
RUN ;
ODS PDF FILE = "c:\temp\marché par constructeur US.pdf" ;
OPTION NOBYLINE ;
TITLE1 "Modèles de la marque #BYVAL1" ;
PROC PRINT DATA = work.voitures (WHERE=(americaine="1"))
           LABEL NOOBS ;
       VAR modele type cylindree puissance transmission ;
       BY constructeur ;
       PAGEBY constructeur ;
RUN ;
OPTION BYLINE ;
TITLE ;
ODS PDF CLOSE ;
```

1. Voir par exemple SAS, *Maîtriser SAS Base et SAS Macro, 2e édition*, H. Kontchou & O. Decourt, Dunod 2007.

Modèles de la marque Cadillac

Nom du modèle	Type de véhicule	Cylindrée (L)	Puissance réelle (CV vapeur)	Mode de transmission
DeVille	Berline	4.9	200	traction
Seville	Familiale	4.6	295	traction

Figure 3.1 — Résultat de l'exemple 3.01, fichier MARCHE PAR CONSTRUCTEUR US.PDF

Pour la construction de rapports soignés, l'intérêt de la procédure PRINT est double : lister les données de base, mais également personnaliser la présentation des sorties d'une autre procédure (dont `ODS OUTPUT` aura transcrit les résultats dans une table SAS intermédiaire). En effet, la procédure PRINT est un moyen simple d'intervenir sur les labels, les formats, l'ordre des colonnes – l'alternative étant d'agir en amont sur le modèle tabulaire associé à l'objet ODS, avec la procédure TEMPLATE (cf. le chapitre 5).

Exemple 3.02 – Procédure PRINT et sortie d'une autre procédure

```
ODS OUTPUT position = work.dictionnaire ;
PROC CONTENTS DATA=livre.voitures VARNUM ;
RUN ;
ODS PDF FILE="c:\temp\dictionnaire des données.pdf" COLUMNS=2 ;
PROC PRINT DATA=work.dictionnaire NOOBS LABEL ;
  VAR label variable type ;
RUN ;
ODS PDF CLOSE ;
```

Label	Variable	Type
Constructeur du véhicule	constructeur	Char
Nom du modèle	modele	Char
Type de véhicule	type	Char
Prix d'entrée de gamme (FF)	prix_min	Num
Prix max - Prix min / 2 (FF)	prix_med	Num
Prix du haut de gamme (FF)	prix_max	Num
Consommation en ville (L aux 100	conso_ville	Num

Figure 3.2 — Résultat de l'exemple 3.02, fichier DICTIONNAIRE DES DONNEES.PDF

3.1.2 Les options de style pour modifier l'aspect du listing

Les options de style peuvent figurer à plusieurs endroits : dans l'instruction PROC PRINT, ou dans les instructions VAR et SUM. Si elles ne sont pas dans la première, elles ne concernent que les colonnes citées dans l'instruction dans laquelle elles figurent. On peut démultiplier les instructions VAR pour obtenir des mises en formes différentes par blocs de variables.

```
PROC PRINT DATA=tableSAS STYLE(zone)=[attributs=valeurs] … ;
  VAR|SUM variable(s) / STYLE(zone)=[attributs=valeurs] ;
  …
RUN ;
```

Dans l'instruction VAR, les zones peuvent être HEADER (en-tête de colonne) ou DATA (valeurs de la variable) ; sans cette précision, c'est l'ensemble de la colonne de listing qui est mise en forme ainsi. Dans l'instruction SUM, la zone peut être HEADER, DATA, TOTAL (pour les sous-totaux) ou GRANDTOTAL (pour le total en toute fin de listing).

Dans l'instruction PROC PRINT, les zones peuvent être HEADER, DATA, TOTAL, GRANDTOTAL, OBS (pour la colonne des numéros d'observation), OBSHEADER (la cellule d'entête de cette colonne), N (la ligne contenant les nombres d'observations) ou encore BYLINE (la ligne de rupture éditée à chaque changement de bloc BY) ; sans précision de zone, la mise en forme vaut pour l'ensemble du listing.

Entre les crochets de l'option de style, on trouve un ou plusieurs couples d'attributs de style et de leurs valeurs, séparés par des espaces. Les principaux attributs utiles à la mise en forme sont présentés dans le tableau 3.1 (repris du tableau 2.3).

Tableau 3.1 — Principaux éléments de style pour la mise en forme

Nom de l'élément	Valeurs possibles	Agit sur…
BACKGROUND	code couleur (cf. section 2.2.1)	Couleur de fond
FOREGROUND	code couleur (cf. section 2.2.1)	Couleur de la police
PROTECTSPECIALCHARS	ON, OFF	Transformation des caractères ayant une signification particulière dans le code généré par l'ODS, par exemple < et > en HTML. Désactiver si on intègre du code HTML dans la sortie
FLYOVER	texte	Info-bulle (HTML) ou note (PDF) ; pas supporté en RTF
URL	adresse de fichier (relative ou absolue)	Lien hypertexte
JUST	LEFT, RIGHT, CENTER	Justification horizontale
VJUST	TOP, BOTTOM, MIDDLE	Justification verticale
FONT_FACE	nom de police entre guillemets	Police utilisée (pour un document PDF, attention à avoir déclaré la police avec la proc FONTREG ; cf. section 1.2.5)
FONT_SIZE	de 1 à 7, ou xPT avec x la taille en points	Taille de la police utilisée (pour le HTML, de 1 pour la plus petite à 7 pour la plus grande)
FONT_STYLE	ITALIC, ROMAN	Police italique ou droite
FONT_WEIGHT	BOLD, MEDIUM, LIGHT	Police grasse, normale ou fine
PREIMAGE ou POSTIMAGE	adresse d'un fichier image GIF ou JPEG (relative de préférence en HTML, absolue en RTF et PDF)	Insertion d'une image avant ou après le texte
ASIS	ON, OFF	Affichage des blancs situés à la gauche du texte (ASIS=ON) ; par défaut, ASIS=OFF

Exemple 3.03 – Procédure PRINT et mise en forme fixe

```
ODS PDF FILE="c:\temp\modèles.pdf"
       STYLE=JOURNAL ;
PROC PRINT DATA=livre.voitures
           NOOBS
           LABEL
           STYLE(DATA)=[BACKGROUND=GRAYA0] ;
  VAR constructeur modele /
            STYLE(DATA)=[FONT_SIZE=14PT] ;
  VAR puissance ;
  VAR prix_min prix_max /
            STYLE(HEADER)=[FLYOVER="En francs 1993"] ;
RUN ;
ODS PDF CLOSE ;
```

Constructeur du véhicule	Nom du modèle	Puissance réelle (CV vapeur)	Prix d'entrée de gamme (FF)	Prix du haut de gamme (FF)
Buick	LeSabre	170	119400	130200
Buick	Roadmaster	180	135600	149400
Cadillac	DeVille	200	198000	217800
Chevrolet	Caprice	170	108000	117600
Chrylser	Concorde	153	110400	110400
Chrysler	Imperial	147	177000	177000
Eagle	Vision	214	105000	127200

En francs 1993

Figure 3.3 — Résultat de l'exemple 3.03, fichier MODELES.PDF

3.1.3 La mise en forme conditionnelle : styles et formats

Il est possible de fournir aux attributs de style des valeurs conditionnées par le contenu des cellules associées : pour cela, il faut avoir créé un format au préalable, qui associe valeur de cellule et valeur de l'attribut de style. Dans la procédure PRINT elle-même, on indiquera alors `attribut=nomFormat.` au lieu de `attribut=valeur` dans l'option `STYLE`.

Exemple 3.04 – Procédure PRINT et mise en forme conditionnelle

```
PROC FORMAT ;
  VALUE couleursPuissance
    LOW-100="GREEN" 100<-200="ORANGE" 200<-HIGH="RED" ;
RUN ;
ODS PDF FILE="c:\temp\modèles.pdf" STYLE=JOURNAL ;
PROC PRINT DATA=livre.voitures NOOBS LABEL ;
    VAR constructeur modele / STYLE(DATA)=[FONT_SIZE=14PT] ;
    VAR puissance / STYLE(DATA)=[BACKGROUND=couleursPuissance.] ;
    VAR prix_min prix_max ;
RUN ;
ODS PDF CLOSE ;
```

Constructeur du véhicule	Nom du modèle	Puissance réelle (CV vapeur)	Prix d'entrée de gamme (FF)	Prix du haut de gamme (FF)
Buick	LeSabre	170	119400	130200
Buick	Roadmaster	180	135600	149400
Cadillac	DeVille	200	198000	217800
Chevrolet	Caprice	170	108000	117600
Chrylser	Concorde	153	110400	110400
Chrysler	Imperial	147	177000	177000
Eagle	Vision	214	105000	127200
Ford	Crown_Victoria	190	120600	130200
Lincoln	Town_Car	210	206400	226800
Oldsmobile	Eighty-Eight	170	117000	131400
Pontiac	Bonneville	170	116400	176400
Acura	Integra	140	77400	112800
Dodge	Colt	92	47400	63600

Figure 3.4 — Résultat de l'exemple 3.04, fichier MODELES.PDF

3.2 LES TABLEAUX CROISÉS AVEC LA PROCÉDURE TABULATE

La procédure TABULATE permet de présenter de nombreuses statistiques (moyennes, sommes, quantiles, pourcentages et comptages) sous forme de tableaux croisés. Si la syntaxe de cette procédure emprunte en partie à la logique de la procédure MEANS (pour les instructions `CLASS` et `VAR`), on retrouve dans l'instruction `TABLE` une logique plus spécifique, et parfois complexe : c'est là qu'on indique l'organisation du tableau proprement dit.

3.2.1 La syntaxe de base de la procédure TABULATE

On indique dans la première instruction quelle est la table SAS lue, *via* la très classique option `DATA`. Les instructions suivantes, `CLASS` et `VAR`, sont optionnelles, mais l'une des deux au moins doit figurer dans la procédure. L'instruction `VAR` énumère les variables numériques sur lesquelles sont calculées des statistiques (moyennes, sommes, etc.). L'instruction `CLASS` indique les autres variables présentes dans le tableau à construire, pour constituer des lignes et des colonnes. En l'absence d'instruction `VAR`, le tableau ne peut comporter comme statistiques que des comptages et des pourcentages.

```
PROC TABULATE DATA = tableSAS ;
  VAR variable(s)Numérique(s)PourStatistiques ;
```

```
  CLASS variable(s)LignesEtColonnes ;
  TABLE … ;
RUN ;
```

L'instruction TABLE indique la disposition des différents éléments du tableau. On y trouve :

- des noms de variables indiquées dans CLASS ou VAR ;
- des parenthèses pour grouper des éléments ;
- des *statistiques* comme les classiques MEAN, SUM, MIN, MAX, N et NMISS, STD et VAR, les quantiles P1, P5, P10, Q1, MEDIAN, Q3, P90, P95 et P99 ainsi que l'écart interquartiles QRANGE, le coefficient de variation CV, l'erreur-type STDERR, l'étendue de la distribution RANGE, les statistiques de nullité de la moyenne T et PROBT ainsi que, à partir de la version 9, les bornes basse et haute d'un intervalle de confiance à 95 % (LCLM et UCLM) ;
- le mot-clé ALL, un récapitulatif en ligne ou en colonne ;
- des chaînes de caractères indiquant le texte à afficher dans une cellule ;
- une ponctuation extrêmement importante (virgule, espace, astérisque).

La ponctuation dans TABLE permet d'indiquer comment les différents éléments fonctionnent les uns avec les autres : la virgule sépare les lignes des colonnes, l'astérisque marque une imbrication et l'espace une juxtaposition (figure 3.5).

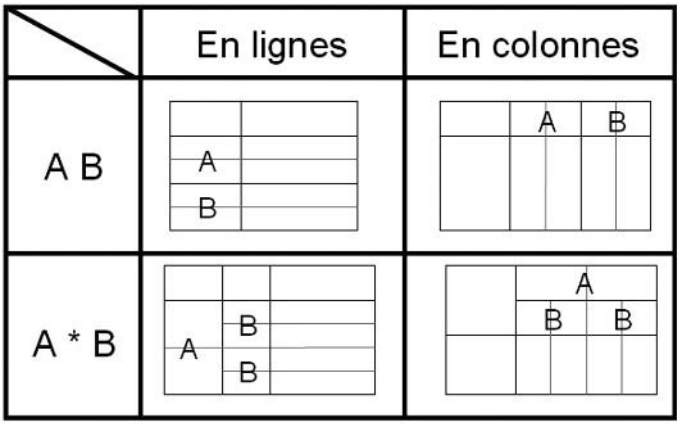


Figure 3.5 — Juxtaposition et imbrication dans une procédure TABULATE

Les tableaux simples

La syntaxe, même basique, de la procédure TABULATE n'étant pas intuitive, nous allons la présenter par l'exemple, en observant comment les mêmes éléments d'un tableau peuvent être permutés et disposés.

Exemple 3.05 – Procédure TABULATE : syntaxe de base

```
PROC TABULATE DATA=livre.voitures ;
       CLASS type ;
       VAR puissance ;
       TABLE type="",
             puissance="Puissance" * (MEAN="Moyenne" MEDIAN="Médiane");
RUN ;
```

Le seul élément en ligne dans l'exemple 3.05 est le type de véhicule. En colonne, on retrouve la puissance du moteur, sur laquelle sont calculées une moyenne et une médiane. La dépendance entre ces statistiques et la variable puissance se matérialise par un astérisque ; la juxtaposition des deux statistiques dans deux colonnes côte à côte est marquée par l'espace qui les sépare. Pour chaque statistique, un libellé entre guillemets indique le texte figurant dans la cellule correspondante.

	Puissance	
	Moyenne	Médiane
Berline	179.45	170.00
Citadine	91.00	90.00
Compacte	131.00	132.00
Familiale	173.09	169.00
Monospace	149.44	151.00
Sportive	160.14	147.50

Figure 3.6 — Résultat de l'exemple 3.05 au format PDF

Exemple 3.06 – Procédure TABULATE : croisements et juxtapositions

```
PROC TABULATE DATA=livre.voitures ;
  CLASS type ;
  VAR puissance poids ;
  LABEL puissance="Puissance" poids="Poids (tonnes)" ;
  TABLE type="", (MEAN="Moyenne" MEDIAN="Médiane") * (puissance poids) ;
  TABLE type="" * (MEAN="Moyenne" MEDIAN="Médiane"), (puissance poids) ;
  TABLE type="" * (puissance poids), (MEAN="Moyenne" MEDIAN="Médiane") ;
RUN ;
```

Dans l'exemple 3.06, on ajoute le poids du véhicule au tableau. On souhaite toujours obtenir une moyenne et une médiane de la puissance, mais aussi du poids ; il y a donc quatre statistiques à présenter pour chaque type de véhicule.

L'écriture `type,(puissance poids)*(MEAN MEDIAN)` affiche ces statistiques dans quatre colonnes juxtaposées (les statistiques apparaissent toutes *après la virgule*) comme on le constate sur la figure 3.7. Tandis que `type*(MEAN MEDIAN),(puissance poids)` affiche deux colonnes, une de poids et une de puissances, et pour type de

véhicule, une ligne de moyennes et une de médianes : c'est ce qu'on retrouve sur la figure 3.8. Enfin, `type*(puissance poids),(MEAN MEDIAN)` affiche une colonne de moyennes, une colonne de médianes, et pour chaque type de véhicule deux lignes, une correspondant aux statistiques sur la puissance et l'autre sur les poids (figure 3.9).

	Moyenne		Médiane	
	Puissance	Poids (tonnes)	Puissance	Poids (tonnes)
Berline	179.45	1.67	170.00	1.62
Citadine	91.00	1.05	90.00	1.06
Compacte	131.00	1.32	132.00	1.35
Familiale	173.09	1.54	169.00	1.57
Monospace	149.44	1.74	151.00	1.69
Sportive	160.14	1.31	147.50	1.29

Figure 3.7 — Résultat de l'exemple 3.06 au format PDF : instruction TABLE n°1

		Puissance	Poids (tonnes)
Berline	Moyenne	179.45	1.67
	Médiane	170.00	1.62
Citadine	Moyenne	91.00	1.05
	Médiane	90.00	1.06
Compacte	Moyenne	131.00	1.32
	Médiane	132.00	1.35
Familiale	Moyenne	173.09	1.54
	Médiane	169.00	1.57
Monospace	Moyenne	149.44	1.74
	Médiane	151.00	1.69
Sportive	Moyenne	160.14	1.31
	Médiane	147.50	1.29

Figure 3.8 — Résultat de l'exemple 3.06 au format PDF : instruction TABLE n°2

		Moyenne	Médiane
Berline	Puissance	179.45	170.00
	Poids (tonnes)	1.67	1.62
Citadine	Puissance	91.00	90.00
	Poids (tonnes)	1.05	1.06
Compacte	Puissance	131.00	132.00
	Poids (tonnes)	1.32	1.35
Familiale	Puissance	173.09	169.00
	Poids (tonnes)	1.54	1.57
Monospace	Puissance	149.44	151.00
	Poids (tonnes)	1.74	1.69
Sportive	Puissance	160.14	147.50
	Poids (tonnes)	1.31	1.29

Figure 3.9 — Résultat de l'exemple 3.06 au format PDF : instruction TABLE n°3

On peut également faire figurer dans les tableaux des récapitulatifs (des sous-totaux si les statistiques sont des sommes ou des comptages) à l'aide du mot-clé `ALL`. L'exemple 3.07 reprend le tableau précédent, dans sa forme à cinq colonnes, pour ajouter une ligne récapitulative. On y trouve moyennes et médianes pour tous les types de voitures confondus sur la figure 3.10.

Exemple 3.07 – Procédure TABULATE : option ALL

```
PROC TABULATE DATA=livre.voitures ;
       CLASS type ;
       VAR puissance poids ;
       TABLE (type="" ALL="Ensemble"),
             (MEAN="Moyennes" MEDIAN="Médianes")*
          (puissance="Puissance" poids="Poids (tonnes)");
RUN ;
```

	Moyennes		Médianes	
	Puissance	Poids (tonnes)	Puissance	Poids (tonnes)
Berline	179.45	1.67	170.00	1.62
Citadine	91.00	1.05	90.00	1.06
Compacte	131.00	1.32	132.00	1.35
Familiale	173.09	1.54	169.00	1.57
Monospace	149.44	1.74	151.00	1.69
Sportive	160.14	1.31	147.50	1.29
Ensemble	143.83	1.39	140.00	1.38

Figure 3.10 — Résultat de l'exemple 3.07 au format PDF

Le contenu de la case supérieure gauche se contrôle avec l'option `BOX`. Cette option s'ajoute après un slash `/`, à la fin de l'instruction `TABLE`.

Exemple 3.08 – Procédure TABULATE : option BOX

```
PROC TABULATE DATA=livre.voitures
    (WHERE=(americaine="1"
        AND constructeur =: "C")) ;
    CLASS type constructeur transmission ;
       VAR puissance ;
    TABLE (constructeur="" ALL="Ensemble") * type="",
           transmission="" * puissance="" * MEAN="Moyenne"
    / BOX="Puissance en CV" ;
RUN ;
```

Puissance en CV		4x4	propulsion	traction
		Moyenne	Moyenne	Moyenne
Cadillac	Berline	.	.	200.00
	Familiale	.	.	295.00
Chevrolet	Berline	.	170.00	.
	Compacte	.	.	110.00
	Familiale	.	.	110.00
	Monospace	165.00	.	170.00
	Sportive	.	230.00	.
Chrysler	Berline	.	.	150.00
	Compacte	.	.	141.00
Ensemble				
	Berline	.	170.00	166.67
	Compacte	.	.	120.33
	Familiale	.	.	202.50
	Monospace	165.00	.	170.00
	Sportive	.	230.00	.

Figure 3.11 — Résultat de l'exemple 3.08 au format PDF

La gestion des valeurs manquantes et des valeurs absentes

On constate sur le résultat de l'exemple 3.08 que les croisements n'existant pas dans les données sont soit remplis de valeurs manquantes, soit masqués quand il s'agit de croisements de variables présentes dans la même dimension (ici, en ligne : il n'y a pas de Chrysler familiale dans les données, donc le tableau n'en comporte pas non plus).

Pour les valeurs manquantes affichées dans le tableau, on peut les remplacer par n'importe quel texte via l'option `MISSTEXT`, insérée à la fin de l'instruction `TABLE`.

Exemple 3.09 – Procédure TABULATE : option MISSTEXT

```
PROC TABULATE DATA=livre.voitures
    (WHERE=(americaine="1"
        AND constructeur =: "C")) ;
    CLASS type constructeur transmission ;
       VAR puissance ;
    TABLE (constructeur="" ALL="Ensemble") * type="",
           transmission="" * puissance="" * MEAN="Moyenne"
    / BOX="Puissance en CV" MISSTEXT="N/A" ;
RUN ;
```

Puissance en CV		4x4	propulsion	traction
		Moyenne	Moyenne	Moyenne
Cadillac	Berline	N/A	N/A	200.00
	Familiale	N/A	N/A	295.00
Chevrolet	Berline	N/A	170.00	N/A
	Compacte	N/A	N/A	110.00
	Familiale	N/A	N/A	110.00
	Monospace	165.00	N/A	170.00
	Sportive	N/A	230.00	N/A
Chrysler	Berline	N/A	N/A	150.00
	Compacte	N/A	N/A	141.00
Ensemble				
	Berline	N/A	170.00	166.67
	Compacte	N/A	N/A	120.33
	Familiale	N/A	N/A	202.50
	Monospace	165.00	N/A	170.00
	Sportive	N/A	230.00	N/A

Figure 3.12 — Résultat de l'exemple 3.09 au format PDF

Concernant les croisements absents des données, on utilisera l'option `PRINTMISS`, à insérer comme `BOX` et `MISSTEXT` en fin d'instruction `TABLE`. `PRINTMISS` parcourt les données pour recenser les valeurs des variables mentionnées dans `CLASS` et prévoit des lignes et des colonnes pour tous les croisements possibles.

Ici, par rapport à la sortie de l'exemple 3.09 (figure 3.12), la ligne « Chrysler familiale » apparaît sur la figure 3.13, bien qu'aucune observation de la table n'y corresponde.

Exemple 3.10 – Procédure TABULATE : option PRINTMISS

```
PROC TABULATE DATA=livre.voitures
    (WHERE=(americaine="1"
        AND constructeur =: "C")) ;
    CLASS type constructeur transmission ;
       VAR puissance ;
    TABLE (constructeur="" ALL="Ensemble") * type="",
          transmission="" * puissance="" * MEAN="Moyenne"
    / BOX="Puissance en CV" MISSTEXT="N/A" PRINTMISS ;
RUN ;
```

Puissance en CV		4x4	propulsion	traction
		Moyenne	Moyenne	Moyenne
Cadillac	Berline	N/A	N/A	200.00
	Compacte	N/A	N/A	N/A
	Familiale	N/A	N/A	295.00
	Monospace	N/A	N/A	N/A
	Sportive	N/A	N/A	N/A
Chevrolet	Berline	N/A	170.00	N/A
	Compacte	N/A	N/A	110.00
	Familiale	N/A	N/A	110.00
	Monospace	165.00	N/A	170.00
	Sportive	N/A	230.00	N/A
Chrysler	Berline	N/A	N/A	150.00
	Compacte	N/A	N/A	141.00
	Familiale	N/A	N/A	N/A
	Monospace	N/A	N/A	N/A

Figure 3.13 — Résultat de l'exemple 3.10 au format PDF

Il est également possible d'aller rechercher les valeurs possibles d'une variable de `CLASS` dans un format (option `PRELOADFMT`) ou dans une table SAS (option `CLASSDATA`).

L'option `PRELOADFMT` suppose qu'une des variables de `CLASS` au moins est associée à un format personnalisé (défini par l'utilisateur via une procédure FORMAT). On ajoute cette option après un slash / en fin d'instruction `CLASS`. Il faut *absolument* l'utiliser en conjonction avec `PRINTMISS`. La définition du format servira à indiquer l'ensemble des valeurs possibles, et tous les croisements les concernant seront prévus, même pour des valeurs absentes des données.

On peut aussi indiquer dans l'instruction CLASS les options PRELOADFMT et EXCLUSIVE. Dans ce cas, les seules valeurs affichées dans le tableau seront celles présentes explicitement dans le format.

Exemple 3.11 – Procédure TABULATE : option PRELOADFMT

```
PROC FORMAT ;
       VALUE $type
       "Familiale","Monospace","Berline"="Grandes"
       "Camion","Tracteur"              ="Utilitaires"
       "Citadine","Compacte"            ="Petites"
       "Sportive"                       ="Sportives" ;
RUN ;
```

```
PROC TABULATE DATA=livre.voitures
       (WHERE=(americaine="1")) ;
       CLASS type transmission / PRELOADFMT ;
       FORMAT type $type. ;
       TABLE type=""*transmission="", N="Nombre de modèles"
              / BOX="Catégorie" PRINTMISS MISSTEXT="0" ;
RUN ;
```

Catégorie		Nombre de modèles
Grandes	4x4	3
	propulsion	5
	traction	18
Utilitaires	4x4	0
	propulsion	0
	traction	0
Petites	4x4	0
	propulsion	0
	traction	14
Sportives	4x4	2
	propulsion	4
	traction	2

Figure 3.14 — Résultat de l'exemple 3.11 au format PDF

L'option `CLASSDATA` s'insère, elle, dans l'instruction `PROC TABULATE`. Elle indique une table SAS dans laquelle toutes les variables de l'instruction `CLASS` sont présentes, et où chaque observation correspond à un croisement de valeurs de ces variables.

Couplés à l'option `EXCLUSIVE` (à indiquer également dans l'instruction `PROC TABULATE`), des croisements présents dans les données mais absents de `CLASSDATA` ne seront pas affichés (et pas comptabilisés dans les récapitulatifs `ALL`). À noter que l'option `CLASSDATA` et l'option `PRINTMISS` ne sont pas compatibles.

Exemple 3.12 – Procédure TABULATE : options CLASSDATA et EXCLUSIVE

```
DATA work.moteurs ;
       INPUT cylindres ;
CARDS ;
4
6
8
12
```

```
;
RUN ;
PROC PRINT DATA = work.moteurs NOOBS ;
RUN ;
PROC TABULATE DATA = livre.voitures
         CLASSDATA = work.moteurs EXCLUSIVE ;
       CLASS cylindres ;
       TABLE cylindres="", N = "Nb de modèles"
           / BOX="Nb de cylindres" MISSTEXT="0" ;
RUN ;
```

cylindres
4
6
8
12

Figure 3.15 — Table MOTEURS

Nb de cylindres	Nb de modèles
4	49
6	31
8	7
12	0

Figure 3.16 — Résultat de l'exemple 3.12 au format PDF

3.2.2 Les pourcentages : comment les calculer ?

Outre les statistiques descriptives usuelles (moyenne, somme, quantiles, comptage), la procédure TABULATE propose également le calcul de pourcentages. Ceux-ci sont associés à deux séries de trois mots-clés : `PCTN`, `ROWPCTN` et `COLPCTN` d'une part, `PCTSUM`, `ROWPCTSUM` et `COLPCTSUM` d'autre part[1].

Dans la première série, il s'agit de pourcentages calculés en fonction du nombre d'observations (d'où le `N` final). Dans la seconde série, ce sont des pourcentages calculés sur le total d'une variable mentionnée dans l'instruction `VAR`. Il peut aussi bien

1. On trouve aussi `PAGEPCTN` et `PAGEPCTSUM` pour le cas (non développé ici) où l'instruction `TABLE` inclurait une variable de page.

s'agir de variables à totaliser (comme des ventes, par exemple) que d'une pondération si l'on présente les résultats d'une enquête ou d'un sondage.

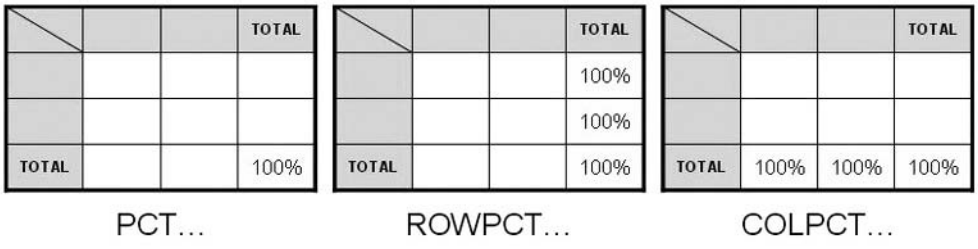


Figure 3.17 — Différents types de pourcentages (PCT, ROWPCT, COLPCT)

L'utilisation de la première ou de la seconde série est identique : le premier pourcentage (`PCT...`) est calculé sur l'ensemble des observations du tableau, le deuxième (`ROWPCT...`) sur les observations consignées dans cette ligne du tableau et le troisième (`COLPCT...`) sur les observations associées à cette colonne du tableau. Un moyen simple de se repérer entre les trois types de pourcentages est de se demander à quel endroit le total fait 100 % (figure 3.17). D'ailleurs, pour faciliter la lecture du tableau, il est recommandé de prévoir des récapitulatifs (`ALL`) en ligne et en colonne pour voir apparaître ces 100 %.

Exemple 3.13 – Procédure TABULATE : PCTN, COLPCTN et ROWPCTN

```
PROC FORMAT ;
  VALUE $pays
      "1" = "USA"
      "0" = "Reste du monde"
  ;
RUN ;
PROC TABULATE DATA = livre.voitures ;
  CLASS transmission americaine ;
  FORMAT americaine $pays. ;
  TABLE (transmission="" ALL="TOTAL")*
        (N="Nb" PCTN="% du total"
         ROWPCTN="% de la ligne" COLPCTN="% de la colonne"),
        (americaine="" ALL="TOTAL") ;
RUN ;
```

		Reste du monde	USA	TOTAL
4x4	Nb	5	5	10
	% du total	5.38	5.38	10.75
	% de la ligne	50.00	50.00	100.00
	% de la colonne	11.11	10.42	10.75
propulsion	Nb	7	9	16
	% du total	7.53	9.68	17.20
	% de la ligne	43.75	56.25	100.00
	% de la colonne	15.56	18.75	17.20
traction	Nb	33	34	67
	% du total	35.48	36.56	72.04
	% de la ligne	49.25	50.75	100.00
	% de la colonne	73.33	70.83	72.04
TOTAL	Nb	45	48	93
	% du total	48.39	51.61	100.00
	% de la ligne	48.39	51.61	100.00
	% de la colonne	100.00	100.00	100.00

Figure 3.18 — Résultat de l'exemple 3.13 au format PDF

Exemple 3.14 – Procédure TABULATE : PCTSUM

```
PROC FORMAT ;
   VALUE $pays "1" = "Made in USA" "0" = "Made in Reste du monde" ;
RUN ;
PROC TABULATE DATA = livre.voitures ;
   CLASS americaine ;
   VAR ventes ;
   FORMAT americaine $pays. ;
   TABLE (americaine="" ALL="TOTAL"),
         ventes=""*(SUM="Total" PCTSUM="% du total") /
         BOX="Ventes annuelles" ;
RUN ;
```

Ventes annuelles	Total	% du total
Made in Reste du monde	69556.00	49.45
Made in USA	71111.00	50.55
TOTAL	140667.00	100.00

Figure 3.19 — Résultat de l'exemple 3.14 au format PDF

Si les mots-clés précédents ne sont pas assez précis, on peut également utiliser PCTN et PCTSUM dans la syntaxe PCTN<variableAuDénominateur>. On indique entre les signes < et > une variable de l'instruction CLASS présente dans le tableau : le pourcentage vaut alors 100 % au niveau d'un récapitulatif ALL qui serait associé à cette variable.

Cette écriture ne s'avère nécessaire que lorsque l'on imbrique plusieurs variables de CLASS (en ligne et/ou en colonne) et qu'on souhaite calculer des pourcentages selon la variable la plus imbriquée.

Exemple 3.15 – Procédure TABULATE : PCTN et dénominateur explicite

```
PROC TABULATE DATA = livre.voitures ;
  WHERE type IN ("Berline","Sportive") ;
  CLASS type transmission ;
  TABLE type="" * (transmission="" ALL="Total"),
        PCTN<transmission ALL>="% dans la catégorie" / BOX="Nb de modèles" ;
RUN ;
```

Nb de modèles		% dans la catégorie
Berline	propulsion	36.36
	traction	63.64
	Total	100.00
Sportive		
	4x4	14.29
	propulsion	35.71
	traction	50.00
	Total	100.00

Figure 3.20 — Résultat de l'exemple 3.15 au format PDF

3.2.3 La mise en forme

Le format choisi par défaut pour les cellules calculées comporte deux décimales, et le séparateur décimal est un point. Quand l'ODS HTML, PDF ou RTF est activé, les comptages apparaissent sans décimales, alors qu'ils en ont dans la fenêtre Output (ODS LISTING). On peut dans tous les cas modifier ce format, soit au niveau de l'ensemble des cellules calculées du tableau (option FORMAT= dans l'instruction PROC TABULATE), soit au niveau d'une statistique en particulier, en la faisant suivre dans l'instruction TABLE d'un astérisque et de FORMAT=.

```
PROC TABULATE DATA = tableSAS FORMAT = nomFormatGlobal. ;
  TABLE … * statistique * FORMAT = nomFormatSpécifique. ;
RUN ;
```

Exemple 3.16 – Procédure TABULATE : formats et cellules calculées

```
PROC TABULATE DATA = livre.voitures ;
   CLASS type ;
   TABLE type="" ALL="TOTAL",
         N="Nb de modèles"*FORMAT=12.
         PCTN="%"*FORMAT=NUMX12.2 ;
RUN ;
```

	Nb de modèles	%
Berline	11	11,83
Citadine	21	22,58
Compacte	16	17,20
Familiale	22	23,66
Monospace	9	9,68
Sportive	14	15,05
TOTAL	93	100,00

Figure 3.21 — Résultat de l'exemple 3.16 au format PDF

On peut également appliquer des options de style au tableau : on retrouvera les mêmes attributs que pour la procédure PRINT, décrits dans le tableau 3.1. Leurs valeurs peuvent être fixes ou conditionnelles à travers un format, comme dans la procédure Print. Ils peuvent être mentionnés à plusieurs emplacements avec l'option `STYLE`, selon le type de cellule à mettre en forme :

- dans l'instruction `PROC TABULATE` s'il s'agit d'une mise en forme commune à toutes les cellules calculées ;
- dans l'instruction `CLASS` ou `VAR` s'il s'agit d'une mise en forme concernant la cellule associée à un nom ou label de variable ;
- dans une instruction `CLASSLEV` spécifique s'il s'agit d'une mise en forme concernant les valeurs d'une variable de `CLASS` ;
- dans une instruction `KEYWORD` spécifique s'il s'agit de la mise en forme d'une cellule associée à `ALL` ou à une statistique ;
- dans l'instruction `TABLE` s'il s'agit d'une mise en forme sur une série de cellules calculées.

Exemple 3.17 – Procédure TABULATE : styles

```
PROC TABULATE DATA = livre.voitures ;
  CLASS type ;
  KEYWORD ALL / STYLE=[BACKGROUND=GRAY60] ;
  TABLE type="" ALL="TOTAL"*[STYLE=[BACKGROUND=GRAY60]],
        N="Nb de modèles"*FORMAT=12. PCTN="%"*FORMAT=NUMX12.2 ;
RUN ;
```

	Nb de modèles	%
Berline	11	11,83
Citadine	21	22,58
Compacte	16	17,20
Familiale	22	23,66
Monospace	9	9,68
Sportive	14	15,05
TOTAL	93	100,00

Figure 3.22 — Résultat de l'exemple 3.17 au format PDF

Un attribut de mise en forme particulièrement utile dans le cadre de la procédure TABULATE est `ASIS=ON` ; en activant cette mise en forme, les blancs situés à gauche d'une chaîne de caractère seront affichés tels quels. Avec un petit travail en amont sur les données, on peut ainsi éviter de multiplier les colonnes à gauche du tableau dans le cas de l'imbrication de variables en ligne.

L'idée est identique à celle de l'option existante `INDENT=nombreEspaces` à indiquer à la fin de l'instruction `TABLE` ; malheureusement, cette option n'est utilisable que dans la destination ODS LISTING.

Exemple 3.18 – Procédure TABULATE : INDENT

```
PROC FORMAT ;
   VALUE $pays
            "0" = "Made in Reste du monde"
            "1" = "Made in USA"
   ;
RUN ;
PROC TABULATE DATA = livre.voitures
              (WHERE = (constructeur IN:("A","B","C"))) ;
   CLASS constructeur americaine ;
   FORMAT americaine $pays. ;
   TABLE americaine *
             constructeur,
          N="Nb de modèles"
```

```
     / INDENT=3 ;
RUN ;
```

On voit sur la figure 3.23 qu'au lieu de deux variables imbriquées (AMERICAINE et CONSTRUCTEUR), on a une seule colonne et que l'imbrication se traduit par un décalage (indentation) de trois caractères blancs.

	Nb de modèles
Made in Reste du monde	
Acura	2.00
Audi	2.00
BMW	1.00
Made in USA	
Buick	4.00
Cadillac	2.00
Chevrolet	8.00
Chrysler	3.00

Figure 3.23 — Résultat de l'exemple 3.18 au format PDF

L'exemple 3.19 permet d'obtenir un résultat semblable, transmissible dans n'importe quelle destination ODS, avec l'option de style `ASIS=ON`.

Exemple 3.19 – Procédure TABULATE : équivalent de INDENT avec ASIS=ON

```
PROC FORMAT ;
   VALUE $pays
       "0" = "Made in Reste du monde" "1" = "Made in USA" ;
RUN ;
ODS OUTPUT list = work.frequences ;
PROC FREQ DATA = livre.voitures (WHERE=(constructeur IN:("A","B","C"))) ;
       TABLE americaine*constructeur / LIST ;
       FORMAT americaine $pays. ;
RUN ;
DATA work.frequences ;
   SET work.frequences ;
   BY americaine ;
   LENGTH categ $ 25 ;
   IF FIRST.americaine THEN DO ;
      categ = F_americaine ;
      freq = . ;
      OUTPUT ;
   END ;
   categ = "   "!!F_constructeur ;
   freq = frequency ;
   OUTPUT ;
```

```
RUN ;
PROC TABULATE DATA = work.frequences ORDER=DATA ;
   CLASS categ ;
   CLASSLEV categ / STYLE=[ASIS=ON] ;
   VAR freq ;
   TABLE categ="", freq=""*SUM="Nb de modèles"*FORMAT=5. / MISSTEXT=" " ;
RUN ;
```

	Nb de modèles
Made in Reste du monde	
Acura	2
Audi	2
BMW	1
Made in USA	
Buick	4
Cadillac	2
Chevrolet	8
Chrysler	3

Figure 3.24 — Résultat de l'exemple 3.19 au format PDF

3.3 LES RAPPORTS COMPLEXES AVEC LA PROCÉDURE REPORT

La procédure REPORT est le croisement entre une étape Data (pour la souplesse), une procédure TRANSPOSE (pour la capacité à organiser des données SAS en lignes et en colonnes), une procédure PRINT (pour l'édition ligne à ligne de son résultat) et une procédure MEANS (pour la capacité à calculer des statistiques). C'est une sorte de couteau suisse de SAS. Mais elle est surtout perçue comme un dernier recours : sa syntaxe, sa logique effraient ; c'est vers elle qu'on se tourne quand tout le reste a échoué.

Pourtant ce vilain petit canard n'est pas si affreux que ça : sa logique est assez carrée, et le coût d'entrée dans sa syntaxe n'est pas supérieur à celui d'une procédure TABULATE. Cette partie commence avec des listes simples, puis intégrant des statistiques, pour arriver ensuite à des rapports plus complexes, avec des calculs de nouvelles variables « à la volée ». Enfin, la procédure REPORT propose, en lien avec l'ODS, une immense souplesse dans l'ajustement des styles.

3.3.1 Le casting du rapport – distribuer les rôles

Deux instructions de base commencent la procédure REPORT : COLUMNS et DEFINE. Elles sont le minimum vital pour que le rapport demandé soit bien construit.

```
PROC REPORT DATA = tableSAS NOWD ;
   COLUMNS listeVariables ;
   DEFINE variable1 / rôle  <"label"> <FORMAT=nomFormat.> < NOPRINT > ;
   DEFINE variable2 / rôle  <"label"> <FORMAT=nomFormat.> < NOPRINT > ;
   …
RUN ;
```

L'option NOWD (ou NOWINDOWS) dans la première instruction est quasiment systématiquement ajoutée. Sans elle, la procédure REPORT propose de construire son rapport dans une fenêtre interactive de la session SAS. Cet environnement de développement du rapport ne sera pas détaillé ici.

Les premiers rôles : instructions COLUMNS *et* DEFINE

Une unique instruction COLUMNS indique comment les colonnes s'organisent de gauche à droite du rapport. Dans cette instruction, toutes les variables sont indiquées sans autre précision, comme dans l'instruction VAR de la procédure PRINT. C'est dans la série d'instructions DEFINE que sera précisé le rôle de chacune. Six rôles peuvent être proposés :

- DISPLAY indique une variable dont le listing rendra compte avec autant de lignes que la table lue compte d'observations. *C'est le rôle par défaut des variables de type caractère*.
- ORDER indique que le listing sera trié selon cette variable (par défaut, avec l'ordre croissant de ses valeurs formatées). Quand la même valeur apparaît à deux lignes consécutives du rapport, elle n'est pas répétée.
- GROUP indique que cette variable sert à construire des agrégats statistiques (ce rôle doit son nom à la partie GROUP BY d'une requête SQL, équivalente au CLASS d'une procédure MEANS). Quand la même valeur apparaît à deux lignes consécutives du rapport, elle n'est pas répétée.
- ANALYSIS indique une variable sur laquelle seront calculées des statistiques. *C'est le rôle par défaut des variables numériques* (avec une somme pour statistique par défaut).
- ACROSS indique une variable dont chacune des valeurs constituera une colonne dans le rapport final. On obtient ainsi une sorte de transposition des données. Le détail des rapports avec une variable ACROSS sera étudié plus loin.
- COMPUTED indique une variable qui sera calculée au cours de l'élaboration du rapport et ne doit pas être cherchée dans la table d'origine.

On peut indiquer dans DEFINE un libellé de colonne entre guillemets, un format après le mot-clé FORMAT= et masquer l'affichage de cette colonne avec NOPRINT. Dans ce dernier cas, la variable est disponible pour des calculs, mais pas visible dans le rapport produit sous forme d'une colonne particulière.

Exemple 3.20 – Procédure REPORT : une liste simple

```
PROC REPORT DATA = livre.voitures NOWD ;
       COLUMNS constructeur modele conso_auto ;
       DEFINE constructeur / DISPLAY ;
       DEFINE modele / DISPLAY ;
       DEFINE conso_auto / DISPLAY ;
RUN ;
```

Constructeur du véhicule	Nom du modèle	Consommation sur autoroute (L aux 100 km)
Acura	Integra	7.59
Acura	Legend	9.41
Audi	90	9.05
Audi	100	9.05
BMW	535i	7.84
Buick	LeSabre	8.4

Figure 3.25 — Résultat de l'exemple 3.20 au format PDF

L'aspect du rapport : DISPLAY *vs* ORDER

On obtiendrait une variante de ce listing en donnant à la variable CONSTRUCTEUR un rôle ORDER plutôt que DISPLAY : les répétitions de valeurs ne seraient alors plus affichées.

```
DEFINE variable1 / rôle ORDER|GROUP ORDER=FORMATTED|INTERNAL|FREQ|DATA
< DESCENDING > ;
```

On peut faire figurer dans DEFINE l'option ORDER en indiquant type d'ordre souhaité : FORMATTED dans l'ordre des valeurs affichées, INTERNAL pour un tri selon les valeurs non formatées, FREQ par fréquence décroissante des modalités et DATA pour l'ordre d'apparition dans la table lue ; si on l'assortit de l'option DESCENDING, l'ordre est renversé par rapport au choix par défaut.

Exemple 3.21 – Procédure REPORT : une liste simple avec un rôle ORDER

```
PROC REPORT DATA = livre.voitures NOWD ;
    COLUMNS constructeur modele conso_auto ;
    DEFINE constructeur / ORDER ;
RUN ;
```

Constructeur du véhicule	Nom du modèle	Consommation sur autoroute (L aux 100 km)
Acura	Integra	7.59
	Legend	9.41
Audi	90	9.05
	100	9.05
BMW	535i	7.84
Buick	LeSabre	8.4

Figure 3.26 — Résultat de l'exemple 3.21 au format PDF

3.3.2 Les synthèses, totaux et sous-totaux

La procédure REPORT permet tout à la fois de produire des listings détaillés avec des lignes récapitulatives (comme les totaux de la procédure PRINT, mais avec un éventail bien plus large de statistiques) et des rapports résumant les données en quelques chiffres.

En présence d'au moins une variable avec le rôle `DISPLAY`, les données seront affichées au même niveau de détail que la table lue ; en revanche, si aucune des variables du rapport n'a de rôle `DISPLAY`, ce seront uniquement des statistiques agrégées qui seront présentées dans le rapport.

```
PROC REPORT DATA = tableSAS NOWD ;
   COLUMNS listeVariables ;
   DEFINE variable1 / DISPLAY|GROUP|ORDER ;
   DEFINE variable2 / ANALYSIS statistique < WEIGHT = varPondération > ;
   …
RUN ;
```

Les statistiques demandées avec DEFINE ANALYSIS

On retrouve dans les statistiques proposées par la procédure REPORT une bonne partie des statistiques disponibles dans les procédures MEANS et TABULATE : les classiques `MEAN`, `SUM`, `MIN`, `MAX`, `N` et `NMISS`, `STD` et `VAR`, les quantiles `P1`, `P5`, `P10`, `Q1`, `MEDIAN`, `Q3`, `P90`, `P95` et `P99` ainsi que l'écart interquartiles `QRANGE`, le coefficient de variation `CV`, l'erreur-type `STDERR`, l'étendue de la distribution `RANGE`, les statistiques de nullité de la moyenne `T` et `PROBT` et des pourcentages `PCTN` et `PCTSUM`.

Il n'existe pas, contrairement à la procédure TABULATE, de notion de pourcentages ligne ou colonne. Ici, les pourcentages sont « colonne » : ils peuvent être sommés à 100 % à la fin de chaque bloc de lignes du rapport.

On ne retrouve pas non plus l'intervalle de confiance de la moyenne que propose la procédure TABULATE ; on peut néanmoins le construire, comme on le verra plus loin.

Exemple 3.22 – Procédure REPORT : édition d'une statistique

```
PROC REPORT DATA = livre.voitures NOWD ;
       COLUMNS type conso_auto ;
       DEFINE type / GROUP ;
       DEFINE conso_auto / ANALYSIS MEAN FORMAT=5.2 ;
RUN ;
```

Type de véhicule	Consommation sur autoroute (L aux 100 km)
Berline	8.82
Citadine	6.77
Compacte	7.94
Familiale	8.88
Monospace	10.79
Sportive	8.29

Figure 3.27 — Résultat de l'exemple 3.22 au format PDF

Contrairement à toutes les autres procédures de calculs statistiques de SAS, la procédure REPORT n'utilise pas une instruction de pondération globale, `WEIGHT` ou `FREQ`. Elle permet d'indiquer à chaque statistique, via l'option `WEIGHT=` à indiquer dans chaque instruction `DEFINE … / ANALYSIS`, quelle est la variable de pondération. On peut ainsi afficher conjointement des statistiques pondérées ou non dans le même tableau, ou pondérées différemment.

```
PROC REPORT DATA = tableSAS NOWD ;
   COLUMNS varGroupe1 < varGroupe2 … > varAnalyse, (stat1 stat2 …) ;
   DEFINE varGroupe1 / GROUP ;
   …
   DEFINE varAnalyse / ANALYSIS < WEIGHT = varPondération > ;
RUN ;
```

Si on souhaite plusieurs statistiques de la même variable, on peut l'indiquer en jouant sur l'instruction `COLUMNS`. Dans `COLUMNS`, on indiquera le nom de la variable d'analyse, puis une virgule, et entre parenthèses la liste des statistiques à calculer. Au niveau de l'instruction `DEFINE`, on indiquera uniquement un rôle `ANALYSIS` pour la variable d'analyse.

Comme dans la procédure TABULATE, on peut demander des statistiques sur plusieurs variables, en groupant les variables par des parenthèses dans l'instruction `COLUMNS : (variable1 variable2),(stat1 stat2)`.

Exemple 3.23 – Procédure REPORT : édition de plusieurs statistiques

```
PROC REPORT DATA = livre.voitures NOWD ;
       COLUMNS type conso_auto,(MEAN MEDIAN STD) ;
       DEFINE type / GROUP ;
       DEFINE conso_auto / ANALYSIS ;
RUN ;
```

On peut modifier les labels et les formats des différentes colonnes, afin d'obtenir un rapport réellement sur mesure. Pour cela, on utilisera cette version améliorée de l'exemple 3.23, avec des instructions `DEFINE` pour chaque statistique.

Exemple 3.23 – Procédure REPORT : édition de plusieurs statistiques

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS type conso_auto,(MEAN MEDIAN STD) ;
  DEFINE type       / GROUP ;
  DEFINE conso_auto / ANALYSIS ;
  DEFINE MEAN       / "Moyenne"    FORMAT=5.1 ;
  DEFINE MEDIAN     / "Médiane"    FORMAT=5.1 ;
  DEFINE STD        / "Ecart-type" FORMAT=5.2 ;
RUN ;
```

	Consommation sur autoroute (L aux 100 km)		
Type de véhicule	Moyenne	Médiane	Ecart-type
Berline	8.8	9.1	0.42
Citadine	6.8	7.1	0.95
Compacte	7.9	7.8	0.76
Familiale	8.9	8.9	0.84
Monospace	10.8	10.7	0.72
Sportive	8.3	8.3	1.02

Figure 3.28 — Résultat de l'exemple 3.23 (2e version) au format PDF

Les récapitulatifs BREAK et RBREAK

En plus des statistiques par groupe, il est possible d'insérer dans le rapport des lignes récapitulatives avec les instructions `BREAK` et `RBREAK`. Cette dernière désigne une ligne récapitulative sur l'ensemble des observations, située au début (`RBREAK BEFORE`) ou à la toute fin du listing (`RBREAK AFTER`).

```
PROC REPORT DATA = tableSAS NOWD ;
   COLUMNS varGroupe1 < varGroupe2 … > varAnalyse, (stat1 stat2 …) ;
   DEFINE varGroupe1 / GROUP ;
   …
   DEFINE varAnalyse / ANALYSIS < WEIGHT = varPondération > ;
   BREAK AFTER|BEFORE varGroupe1 / < SUMMARIZE > < PAGE > ;
   RBREAK AFTER|BEFORE / < SUMMARIZE > < PAGE > ;
RUN ;
```

L'instruction BREAK indique qu'on ajoute une ligne récapitulative avant ou après chaque valeur d'une variable de groupe. BREAK n'est donc utile que si les groupes sont définis par au moins deux variables, comme on le voit sur la figure 3.29.

Si l'on souhaite un saut de page à la place (ou en plus) du récapitulatif, on ajoutera à BREAK ou RBREAK l'option PAGE à la place (ou en plus) de SUMMARIZE.

Sans que des précisions en ce sens ne soient nécessaires, les lignes récapitulatives générées par BREAK et RBREAK sont automatiquement affichées en italique dans toutes les destinations ODS sauf OUTPUT et LISTING.

Var1	Var2	Var3	...	
A	1	a		
		b		
A	1			BREAK AFTER **var2**
A	2	a		
		c		
A	2			BREAK AFTER **var2**
A				BREAK AFTER **var1**
B	2	b		
		c		
B	2			BREAK AFTER **var2**

Figure 3.29 — Les récapitulatifs BREAK

Dans les BREAK et RBREAK, on indiquera obligatoirement l'option SUMMARIZE pour demander les calculs ; ceux-ci reprennent les mêmes statistiques qu'indiquées dans DEFINE…ANALYSIS.

Exemple 3.24 – Procédure REPORT : récapitulatifs (BREAK et RBREAK)

```
PROC FORMAT ;
  VALUE $us
    "0" = "Etrangère"
```

```
    "1" = "Américaine"
  ;
RUN ;
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS type americaine (conso_ville poids),(MEAN MEDIAN) ;
  DEFINE type       / GROUP                        ;
  DEFINE americaine / GROUP "Marque"  FORMAT=$us. ;
  DEFINE MEAN       /       "Moyenne" FORMAT=6.2  ;
  DEFINE MEDIAN     /       "Médiane" FORMAT=6.2  ;
  RBREAK BEFORE    / SUMMARIZE ;
  BREAK AFTER type / SUMMARIZE ;
RUN ;
```

		Consommation en ville (L aux 100 km)		Poids du véhicule (tonnes)	
Type de véhicule	**Marque**	**Moyenne**	**Médiane**	**Moyenne**	**Médiane**
		11.05	*11.20*	*1.39*	*1.38*
Berline	Américaine	12.89	12.38	1.67	1.62
Berline		*12.89*	*12.38*	*1.67*	*1.62*
Citadine	Américaine	8.61	8.11	1.07	1.07
	Etrangère	7.93	8.11	1.04	1.06
Citadine		*8.15*	*8.11*	*1.05*	*1.06*
Compacte	Américaine	10.07	10.23	1.26	1.26

Figure 3.30 — Résultat de l'exemple 3.24 au format PDF

L'ajout de récapitulatifs sur mesure

L'édition d'un récapitulatif avec `BREAK` et `RBREAK` se fait avec une mise en forme contrainte. Il est cependant possible de les amender en modifiant les valeurs des variables listées dans un bloc `COMPUTE`, et même d'ajouter des lignes de texte construites sur mesure *via* l'instruction `LINE` dans ce bloc `COMPUTE`.

Un bloc `COMPUTE BEFORE` est exécuté avant l'édition de la première ligne du rapport, tandis que `COMPUTE AFTER` s'exécute après la dernière ligne du rapport. Quand on indique `COMPUTE BEFORE` ou `AFTER` puis le nom d'une variable de rôle `GROUP`, le bloc s'exécute avant (respectivement à la fin) d'un bloc de valeurs de cette variable.

Avant la construction du rapport, toutes les statistiques demandées dans `COLUMNS` ou `DEFINE ANALYSIS` sont calculées, pour toutes les combinaisons des variables `GROUP`, et sur l'ensemble des observations. Ces statistiques sont ensuite disponibles dans les blocs `COMPUTE`, soit sous leur nom pour `N`, soit sous la forme `nomVariable.statistique` pour toutes les autres statistiques.

```
PROC REPORT DATA = tableSAS NOWD ;
  COLUMNS … ;
  DEFINE … ;
  COMPUTE BEFORE|AFTER < varGroupe > ;
    calculs
    LINE "texte" variable.statistique < format. > "texte" … ;
  ENDCOMP ;
RUN ;
```

Les calculs autorisés dans un bloc `COMPUTE` sont globalement les mêmes que dans une étape Data. On peut utiliser des fonctions, créer de nouvelles variables, reprendre les valeurs d'une variable existante… les possibilités sont nombreuses, et plus détaillées dans la section 3.3.3.

L'instruction `LINE`, quant à elle, fonctionne sur le principe d'un `PUT` dans une étape Data : on peut mélanger texte entre guillemets et informations variables (sous la forme indiquée, on a les valeurs des différentes statistiques pour le récapitulatif correspondant au timing du bloc `COMPUTE`) éventuellement suivies d'un format pour afficher les valeurs de manière plus maîtrisée. Pour disposer les différents éléments, on peut uniquement les énoncer à la suite les uns des autres, ou jouer sur des indications de déplacement du curseur d'écriture : `+1` décale d'un caractère vers la droite, `+(-1)` vers la gauche, tandis que `@20` écrit à hauteur du 20e caractère en partant de la gauche. On peut indiquer autant d'instructions `LINE` que l'on souhaite, pour écrire autant de lignes dans le listing final. En revanche, on ne peut pas conditionner l'exécution d'un `LINE` ; même placé dans une syntaxe `IF condition THEN LINE` le texte sera *systématiquement* affiché !

Exemple 3.25 – Procédure REPORT : récapitulatif sur mesure avec LINE

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS type (conso_ville poids),(MEAN MEDIAN) N ;
  DEFINE type / GROUP ;
  DEFINE N / NOPRINT ;
  COMPUTE AFTER ;
    type = "Ensemble" ;
    LINE "Statistiques calculées sur" +1 N 2. +1 "véhicules" ;
  ENDCOMP ;
  RBREAK AFTER / SUMMARIZE ;
RUN ;
```

	Consommation en ville (L aux 100 km)		Poids du véhicule (tonnes)	
Type de véhicule	MEAN	MEDIAN	MEAN	MEDIAN
Berline	12.890909	12.38	1.674	1.617
Citadine	8.1542857	8.11	1.0477619	1.062
Compacte	10.43875	10.23	1.3219375	1.345
Familiale	12.144545	12.38	1.5402273	1.573
Monospace	13.906667	13.84	1.7352222	1.692
Sportive	11.116429	10.46	1.3135714	1.2945
Ensemble	*11.054086*	*11.2*	*1.392043*	*1.377*
Statistiques calculées sur 93 véhicules				

Figure 3.31 — Résultat de l'exemple 3.25 au format PDF

On voit sur la figure 3.31 que l'ordre dans lequel on cite `RBREAK AFTER` et `COMPUTE AFTER` n'a pas d'importance : c'est `RBREAK` qui s'affichera en premier. L'ordre aurait été le même pour afficher en début de listing : `RBREAK BEFORE` s'affichera avant `COMPUTE BEFORE`.

En revanche, l'ordre *d'exécution* est bien `COMPUTE` avant `RBREAK` : on peut ainsi modifier la valeur de la variable TYPE au moment du récapitulatif (dans `COMPUTE`) avant que celui-ci ne soit affiché (`RBREAK`) et donc y faire figurer le texte « ensemble ».

Autre astuce utilisée dans l'exemple 3.25 : comme la statistique `N` n'est pas calculée par défaut, on la demande dans le rapport (via l'instruction `COLUMNS`). Mais on indique dans `DEFINE` qu'on ne l'affichera pas (option `NOPRINT`) ! Elle est donc calculée mais uniquement affichée à travers l'instruction `LINE`.

3.3.3 Les calculs dans la proc Report

On l'a dit, il est très facile de créer de nouvelles variables en cours de rapport. On peut pour cela indiquer des noms de variables inédits dans `COLUMNS`, puis préciser pour lever toute équivoque un rôle `COMPUTED` pour ces variables, et enfin calculer leurs valeurs dans les blocs `COMPUTE`. Au sein de ces blocs, quasiment toute la syntaxe d'une étape Data peut être utilisée :

- l'intégralité des fonctions SAS (sauf `LAG` et `DIF`) et des routines (`CALL`) ;
- les instructions conditionnelles `IF` et `SELECT` ;
- les blocs de variables (`ARRAY`) et leur lecture par boucle (`DO`) ;
- les écritures d'accumulation (`variable + quantité`).

```
PROC REPORT DATA = tableSAS NOWD ;
  COLUMNS …  varComp1 varComp2 ;
  DEFINE varComp1 / COMPUTED ;
  DEFINE varComp2 / COMPUTED ;
  COMPUTE varComp1 ;
    calculs ne pouvant utiliser varComp2
  ENDCOMP ;
  COMPUTE varComp2 ;
    calculs pouvant utiliser varComp1
  ENDCOMP ;
RUN ;
```

On peut utiliser les différentes informations présentes dans le rapport :

- les noms des variables de type `GROUP`, `ORDER`, `ACROSS` ou `DISPLAY` pour utiliser leur valeur courante au moment de l'exécution du bloc `COMPUTE` ;
- les écritures `variable.statistique` pour les statistiques sur les variables `ANALYSIS` ;
- les noms d'autres variables calculées au cours du rapport, à condition qu'elles apparaissent dans `COLUMNS` *à gauche* de celle que l'on est en train de calculer ;
- dans le cas particulier de variables en colonnes (`ACROSS`), on peut utiliser des numéros de colonnes : `_C1_`, `_C2_`, …, numérotées de gauche à droite ;
- une variable automatique du nom de `_BREAK_` indique si l'on se trouve sur une ligne de détail (elle vaut alors `" "`), sur un récapitulatif de groupe (elle vaut alors le nom de la variable de groupe en question) ou sur un récapitulatif global (elle vaut alors `"_RBREAK_"`).

On peut comparer sur les exemples 3.26 et 3.27 les différentes manières d'appeler la valeur d'une variable selon son rôle : tout d'abord par son nom pour une variable de rôle `DISPLAY`, puis par la notation composée pour une statistique.

Exemple 3.26 – Procédure REPORT : bloc COMPUTE et variables calculées (1)

```
PROC REPORT DATA = livre.voitures NOWD ;
       COLUMNS constructeur modele poids puissance rpp ;
       DEFINE constructeur / DISPLAY ;
       DEFINE modele / DISPLAY ;
       DEFINE poids / DISPLAY "Poids du véhicule (kg)" FORMAT=NLNUM6. ;
       DEFINE puissance / DISPLAY ;
       DEFINE rpp / COMPUTED "Rapport poids/puissance" FORMAT=NUMX8.2 ;
       COMPUTE poids ;
              poids = poids * 1000 ;
       ENDCOMP ;
       COMPUTE rpp ;
              rpp = poids / puissance ;
       ENDCOMP ;
RUN ;
```

Dans l'exemple 3.26, on voit qu'on peut également intervenir dans le bloc COMPUTE sur une variable qui existe déjà (ici, POIDS est transformée de tonnes en kilos). Dès lors, ce sera la nouvelle valeur qui sera utilisée dans tous les blocs COMPUTE concernant une variable énoncée à sa droite dans COLUMNS.

Constructeur du véhicule	Nom du modèle	Poids du véhicule (kg)	Puissance réelle (CV vapeur)	Rapport poids puissance
Acura	Integra	1 225	140	8,75
Acura	Legend	1 613	200	8,07
Audi	90	1 529	172	8,89
Audi	100	1 542	172	8,97
BMW	535i	1 649	208	7,93
Buick	LeSabre	1 572	170	9,25
Buick	Roadmaster	1 860	180	10,33
Buick	Century	1 305	110	11,86

Figure 3.32 — Résultat de l'exemple 3.26 au format PDF

Exemple 3.27 – Procédure REPORT : bloc COMPUTE et variables calculées (2)

```
PROC REPORT DATA = livre.voitures NOWD ;
       COLUMNS type poids puissance rpp ;
       DEFINE type / GROUP ;
       DEFINE poids / ANALYSIS MEAN "Poids du véhicule (kg)" FORMAT=5. ;
       DEFINE puissance / ANALYSIS MEAN FORMAT=6.1;
       DEFINE rpp / COMPUTED "Rapport poids/puissance" FORMAT=5.2 ;
       COMPUTE poids ;
              poids.mean = poids.mean * 1000 ;
       ENDCOMP ;
       COMPUTE rpp ;
              rpp = poids.mean / puissance.mean ;
       ENDCOMP ;
RUN ;
```

Type de véhicule	Poids du véhicule (kg)	Puissance réelle (CV vapeur)	Rapport poids puissance
Berline	1674	179.5	9.33
Citadine	1048	91.0	11.51
Compacte	1322	131.0	10.09
Familiale	1540	173.1	8.90
Monospace	1735	149.4	11.61
Sportive	1314	160.1	8.20

Figure 3.33 — Résultat de l'exemple 3.27 au format PDF

L'ordre d'exécution des blocs COMPUTE

À l'exécution de la procédure REPORT, SAS consolide tout d'abord les statistiques et les compteurs définis par `ANALYSIS` et `GROUP`, ainsi que les récapitulatifs, même en l'absence d'instructions `BREAK` ou `RBREAK`.

Puis les lignes du rapport sont constituées une à une, sur le modèle de l'instruction `COLUMNS`. Selon le type de la ligne, on exécutera :

- le bloc `COMPUTE BEFORE` s'il s'agit de la première ligne du rapport ;
- le bloc `COMPUTE AFTER` s'il s'agit de la dernière ligne du rapport ;
- le bloc `COMPUTE BEFORE varGroupe` si on s'apprête à commencer un nouveau bloc de valeurs de la variable VARGROUPE ;
- le bloc `COMPUTE AFTER varGroupe` si on vient de finir un bloc de valeurs de la variable VARGROUPE ;
- les blocs `COMPUTE variable` dans l'ordre des variables telles qu'elles sont citées dans `COLUMNS`.

L'utilisation de variables temporaires

Les variables créées dans un bloc `COMPUTE` et citées nulle part ailleurs sont invisibles dans le rapport final. Elles permettent néanmoins de rendre certains services, puisqu'au contraire des variables de l'instruction `COLUMNS`, elles ne sont pas réinitialisées avant une nouvelle ligne du rapport. On peut donc utiliser ces variables temporaires pour conserver en mémoire une valeur à différentes observations. L'exemple 3.28 illustre cette utilisation dans deux cas : le calcul de pourcentages (on retient le nombre total d'observations en début de listing dans une variable temporaire qui sert ensuite aux calculs de pourcentages ligne par ligne dans le rapport) et la comparaison à la moyenne générale.

Exemple 3.28 – Procédure REPORT : blocs COMPUTE et variables temporaires

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS type conso_ville ecart N pct ;
  DEFINE type        / GROUP ;
  DEFINE conso_ville / ANALYSIS MEAN FORMAT=8.1 ;
  DEFINE ecart       / COMPUTED FORMAT=8.2 "Ecart à la moyenne générale" ;
  DEFINE N           / "Effectifs" ;
  DEFINE pct         / COMPUTED FORMAT=PERCENT12. " ";
  COMPUTE BEFORE ;
    nTotal = N ;
    moyGen = conso_ville.mean ;
  ENDCOMP ;
  COMPUTE ecart ;
    ecart = conso_ville.mean - moyGen ;
  ENDCOMP ;
  COMPUTE pct ;
    pct = N / nTotal ;
  ENDCOMP ;
  RBREAK AFTER / SUMMARIZE ;
RUN ;
```

Type de véhicule	Consommation en ville (L aux 100 km)	Ecart à la moyenne générale	Effectifs	
Berline	12.9	1.84	11	12%
Citadine	8.2	-2.90	21	23%
Compacte	10.4	-0.62	16	17%
Familiale	12.1	1.09	22	24%
Monospace	13.9	2.85	9	10%
Sportive	11.1	0.06	14	15%
	11.1	*0.00*	*93*	*100%*

Figure 3.34 — Résultat de l'exemple 3.28 au format PDF

La création de variables caractère

La création *via* un bloc `COMPUTE` ne pose pas de problème particulier, mais se fait à travers une syntaxe spécifique où on indique la nature de cette nouvelle variable et sa longueur maximale.

```
COMPUTE nomVariable / CHARACTER LENGTH=longueurMaxi ;
  calculs
ENDCOMP ;
```

L'exemple 3.29 crée une variable texte qui contient les bornes d'un intervalle de confiance à 95 % autour de la moyenne. On calcule cet intervalle avec la formule

MOYENNE ± ERREUR-TYPE * 1,96. Par souci de précision, le 1,96 est calculé directement par SAS à partir de la distribution de la loi normale.

On utilise dans cet exemple un *alias de colonne* : dans `COLUMNS`, on indique qu'on aura deux fois la variable PUISSANCE sous deux noms différents, le sien et ET. On peut alors utiliser deux instructions `DEFINE` différentes (dont une `NOPRINT`) pour obtenir toutes les statistiques nécessaires à la construction de l'intervalle de confiance. Les alias ont le même statut qu'une variable calculée, même s'ils n'ont pas besoin d'un bloc `COMPUTE` pour avoir des valeurs.

Dans cet exemple, le code ne fonctionnerait pas si on mentionnait l'alias ET à droite de IC dans `COLUMNS` : on ne pourrait alors pas utiliser sa valeur dans `COMPUTE`.

Exemple 3.29 – Procédure REPORT : blocs COMPUTE et variables caractère

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS type puissance puissance=et ic ;
  DEFINE type / GROUP ;
  DEFINE puissance/ ANALYSIS MEAN FORMAT=5.1 ;
  DEFINE et / ANALYSIS STDERR NOPRINT ;
  DEFINE ic / COMPUTED "Intervalle de confiance à 95% pour la moyenne" ;
  COMPUTE ic / CHARACTER LENGTH=15 ;
    ic = CAT("[",
             PUT(puissance.mean-et*PROBIT(.975), 5.1),
             " ; ",
             PUT(puissance.mean+et*PROBIT(.975), 5.1),
             "]") ;
  ENDCOMP ;
RUN ;
```

Type de véhicule	Puissance réelle (CV vapeur)	Intervalle de confiance à 95% pour la moyenne
Berline	179.5	[166.5 ; 192.4]
Citadine	91.0	[82.0 ; 100.0]
Compacte	131.0	[119.8 ; 142.2]
Familiale	173.1	[151.2 ; 195.0]
Monospace	149.4	[136.9 ; 162.0]
Sportive	160.1	[121.2 ; 199.1]

Figure 3.35 — Résultat de l'exemple 3.29 au format PDF

Les variables de type caractère, même créées lors de la procédure REPORT, ne sont pas incluses dans les calculs des récapitulatifs (SUMMARIZE) des instructions BREAK et RBREAK.

3.3.4 Les transpositions, la mise en colonnes et les statistiques

On peut obtenir avec la procédure REPORT des tableaux croisés dans le même style que ceux proposés par la procédure TABULATE. Pour cela, il faut déclarer la variable apparaissant en colonnes comme ayant le rôle ACROSS. Par défaut, les colonnes sont remplies avec des comptages.

On peut obtenir une colonne récapitulative en demandant en sus la statistique N dans COLUMNS, et une ligne récapitulative avec RBREAK.

Exemple 3.30 – Procédure REPORT : variable ACROSS et comptages

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS constructeur type N ;
  DEFINE constructeur / GROUP ;
  DEFINE type / ACROSS ;
  DEFINE N / "Total" ;
  RBREAK AFTER / SUMMARIZE ;
RUN ;
```

	Type de véhicule						
Constructeur du véhicule	Berline	Citadine	Compacte	Familiale	Monospace	Sportive	Total
Acura	.	1	.	1	.	.	2
Audi	.	.	1	1	.	.	2
BMW	.	.	.	1	.	.	1
Buick	2	.	.	2	.	.	4
Cadillac	1	.	.	1	.	.	2
Chevrolet	1	.	2	1	2	2	8
Chrysler	2	.	1	.	.	.	3
Dodge	.	2	1	1	1	1	6
Eagle	1	1	.	.	.	.	2

Figure 3.36 — Résultat de l'exemple 3.30 au format PDF

Si on souhaite remplir les cases de ce tableau avec des statistiques autres que des comptages, on utilisera la virgule dans l'instruction COLUMNS pour marquer l'imbrication de la statistique dans la dimension colonne.

```
COLUMNS varLigne varColonne, (varAnalyse1 < varAnalyse2 … >);
DEFINE varLigne / GROUP ;
DEFINE varColonne / ACROSS ;
```

```
DEFINE varAnalyse1 / ANALYSIS statistique ;
DEFINE varAnalyse2 / ANALYSIS statistique ;
```

On peut imbriquer plusieurs statistiques d'une même variable en usant d'alias de colonnes (cf. l'exemple 3.29), ou des statistiques de plusieurs variables.

Exemple 3.31 – Procédure REPORT : variable ACROSS et statistiques

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS type transmission, (puissance) ;
  DEFINE type / GROUP ;
  DEFINE transmission / ACROSS " " ;
  DEFINE puissance / ANALYSIS MEAN FORMAT=5.1 "Puissance moyenne";
RUN ;
```

	4x4	propulsion	traction
Type de véhicule	Puissance moyenne	Puissance moyenne	Puissance moyenne
Berline	.	187.5	174.9
Citadine	81.5	.	92.0
Compacte	130.0	122.0	132.5
Familiale	.	213.6	161.2
Monospace	149.0	.	150.0
Sportive	196.0	196.0	124.3

Figure 3.37 — Résultat de l'exemple 3.31 au format PDF

Dans le cadre d'une variable ACROSS, la définition de nouvelles variables avec un bloc COMPUTE est plus complexe. On utilisera plutôt les noms de colonnes du rapport (_C1_, _C2_, etc.) dans les formules de calcul. Attention, il faut compter dans les colonnes celles qui ne sont pas affichées au final (DEFINE NOPRINT).

On le constate sur la figure 3.38, résultat de l'exemple 3.32 dont le but est de calculer des pourcentages ligne ; on y utilise par ailleurs la fonction SAS COALESCE qui renvoie le 1er argument s'il est non manquant, et à défaut le deuxième argument.

Exemple 3.32 – Procédure REPORT : variable ACROSS et bloc COMPUTE

```
PROC REPORT DATA = livre.voitures NOWD ;
       COLUMNS type puissance transmission,(n pct) total ;
       DEFINE type / GROUP ;
       DEFINE puissance / ANALYSIS N NOPRINT ;
       DEFINE transmission / ACROSS ;
       DEFINE pct / COMPUTED FORMAT = PERCENT12.1 ;
       DEFINE total / COMPUTED FORMAT = PERCENT12.1 ;
       COMPUTE BEFORE type ;
              nType = puissance.n ;
```

```
        ENDCOMP ;
        COMPUTE pct ;
               _C4_ = COALESCE(_C3_/nType,0) ;
               _C6_ = COALESCE(_C5_/nType,0) ;
               _C8_ = COALESCE(_C7_/nType,0) ;
        ENDCOMP ;
        COMPUTE total ;
               total = 1 ;
        ENDCOMP ;
RUN ;
```

	Mode de transmission						
	4x4		propulsion		traction		
Type de véhicule	n	pct	n	pct	n	pct	total
Berline	.	0.0%	4	36.4%	7	63.6%	100.0%
Citadine	2	9.5%	.	0.0%	19	90.5%	100.0%
Compacte	1	6.3%	2	12.5%	13	81.3%	100.0%
Familiale	.	0.0%	5	22.7%	17	77.3%	100.0%
Monospace	5	55.6%	.	0.0%	4	44.4%	100.0%
Sportive	2	14.3%	5	35.7%	7	50.0%	100.0%

Figure 3.38 — Résultat de l'exemple 3.32 au format PDF

Si dans l'exemple 3.32, on peut aisément calculer l'ensemble des pourcentages en recopiant la formule, sur des colonnes assez nombreuses, le recours à un ARRAY permet de résumer le code et de le rendre plus aisé à maintenir.

L'exemple 3.33 s'appuie sur une table exemple de SAS, SASHELP.PRDSALE, qui détaille les ventes d'un certain nombre de produits par une société fictive au cours des années 1993 et 1994 dans plusieurs pays. Le but de ce programme est de créer un tableau de bord incluant somme des ventes par trimestre et évolution d'un trimestre à l'autre.

Exemple 3.33 – Procédure REPORT : variable ACROSS et ARRAY

```
PROC REPORT DATA = sashelp.prdsale NOWD ;
 COLUMNS product month,(actual evol) ;
 DEFINE product / GROUP ;
 DEFINE month / ACROSS "Trimestre" FORMAT=YYQRD8. ORDER=INTERNAL ;
 DEFINE evol / COMPUTED "Evolution" FORMAT = PERCENTN12.1 ;
 DEFINE actual / ANALYSIS SUM "Ventes" FORMAT = NLNUM12. ;
 COMPUTE evol ;
  ARRAY colonnes
   _C2_  _C3_  _C4_  _C5_  _C6_  _C7_  _C8_  _C9_
   _C10_ _C11_ _C12_ _C13_ _C14_ _C15_ _C16_ _C17_ ;
   DO i = 1 TO DIM(colonnes) ;
    IF MOD(i,2)=0 /* colonnes paires = évolutions */ THEN DO ;
```

```
      IF i > 2 THEN
         colonnes(i) = (colonnes(i-1)-colonnes(i-3))/colonnes(i-3) ;
    END ;
   END ;
  ENDCOMP ;
  RBREAK AFTER / SUMMARIZE ;
RUN ;
```

Dans l'exemple 3.33, la précision `ORDER=INTERNAL` pour la variable `ACROSS` est indispensable puisqu'on travaille sur une date. Avec le choix par défaut (`ORDER=FORMATTED`), on aurait eu des colonnes par ordre alphabétique des dates telles qu'elles sont affichées. Avec `ORDER=INTERNAL`, l'agencement des colonnes suivra l'ordre chronologique comme on le voit sur la figure 3.39..

	Trimestre									
	1993-I		1993-II		1993-III		1993-IV		1994-I	
Product	Ventes	Evolution	Ventes	Evolution	Ventes	Evolution	Ventes	Evolution	Ventes	Evolu
BED	14 028	.	20 341	45.0%	17 981	-11.6%	17 113	-4.8%	17 521	2
CHAIR	17 100	.	14 867	-13.1%	20 214	36.0%	20 839	3.1%	18 177	-12
DESK	17 977	.	21 140	17.6%	19 013	-10.1%	18 037	-5.1%	18 944	5
SOFA	21 409	.	18 937	-11.5%	20 627	8.9%	18 459	-10.5%	16 818	-8
TABLE	18 756	.	20 518	9.4%	15 599	-24.0%	16 522	5.9%	18 303	10
	89 270	.	*95 803*	*7.3%*	*93 434*	*-2.5%*	*90 970*	*-2.6%*	*89 763*	*-1*

Figure 3.39 — Résultat de l'exemple 3.33 au format PDF

3.3.5 La mise en forme avancée

De nombreux éléments de syntaxe de la procédure REPORT n'ont pas encore été abordés. La plupart d'entre eux permettent surtout de contrôler finement l'aspect des sorties. Pour une bonne part, ces options ne sont appliquées qu'à travers la destination LISTING de l'ODS, et ne sont pas rendues (ou pas correctement) dans les destinations plus « glamour » comme RTF, PDF ou HTML. De telles options ne seront pas détaillées ici.

En revanche, cette section traitera de la manière dont on peut traiter les données manquantes dans un rapport, puis des en-têtes de rapports, et enfin de la mise en forme à destination de l'ODS spécifiquement, avec des liens (pour PDF et HTML) et des styles (couleurs, polices, images, etc.).

Les valeurs manquantes

À l'image de ce que l'on peut faire dans la procédure TABULATE avec des options comme `PRINTMISS`, on peut générer dans la procédure REPORT des croisements de valeurs dans les variables de groupe ou de colonnes qui ne sont pas présents dans les données lues. On utilisera pour cela les options `COMPLETEROWS` et `COMPLETECOLS`.

```
PROC REPORT DATA = tableSAS < MISSING >
                            < COMPLETEROWS > < COMPLETECOLS > ;
```

Par ailleurs, l'option MISSING permet, comme dans les procédures MEANS et TABULATE, de considérer que la valeur manquante doit faire l'objet d'une ligne ou d'une colonne à part entière, et non pas être exclue de tous les calculs.

Exemple 3.34 – Procédure REPORT : option COMPLETEROWS

```
PROC REPORT DATA = livre.voitures NOWD COMPLETEROWS ;
       COLUMNS type transmission n ;
       DEFINE type / GROUP ;
       DEFINE transmission / GROUP ;
RUN ;
```

Type de véhicule	Mode de transmission	n
Berline	4x4	0
	propulsion	4
	traction	7
Citadine	4x4	2
	propulsion	0
	traction	19
Compacte	4x4	1

Figure 3.40 — Résultat de l'exemple 3.34 au format PDF

On retrouve également, mais au niveau des instructions DEFINE, une option déjà vue avec la procédure TABULATE : PRELOADFMT. Ce sont alors toutes les valeurs prévues par le format associé à la variable de l'instruction DEFINE qui seront listées dans le rapport, même si elles n'apparaissent pas dans les données. On peut également se limiter aux valeurs prévues par le format en précisant PRELOADFMT EXCLUSIVE. D'autres valeurs, vues dans les données mais non traitées par le format, ne figureront pas dans le rapport.

```
DEFINE variable / rôle FORMAT=nomFormat. < PRELOADFMT < EXCLUSIVE > > ;
```

Les en-têtes de rapports

L'option SPLIT permet d'indiquer à quel caractère les en-têtes de colonnes doivent être renvoyés à la ligne. On indiquera comme valeur de cette option un caractère peu courant, pour éviter les confusions : tilde, astérisque, dièse…

```
PROC REPORT DATA = tableSAS < SPLIT="caractère" | NOHEADER > ;
```

L'option NOHEADER supprime pour sa part toute édition d'en-têtes dans le rapport.

Exemple 3.35 – Procédure REPORT : option NOHEADER

```
ODS PDF FILE = "c:\temp\report.pdf" COLUMNS=3 ;
PROC FORMAT ;
       VALUE $debut
              "A"-<"F" = "A - E"
              "F"-<"P" = "F - O"
              "P"-<"a" = "P - Z"
       ;
RUN ;
PROC REPORT DATA = livre.voitures NOWD NOHEADER ;
       COLUMNS constructeur=initiale constructeur ;
       DEFINE initiale / GROUP FORMAT=$debut. ;
       DEFINE constructeur / GROUP ;
       BREAK AFTER constructeur / PAGE ;
RUN ;
ODS PDF CLOSE ;
```

A - E	Acura
	Audi
	BMW
	Buick
	Cadillac
	Chevrolet
	Chrysler
	Dodge
	Eagle

F - O	Ford
	Geo
	Honda
	Hyundai
	Infiniti
	Lexus
	Lincoln
	Mazda
	Mercedes-Benz
	Mercury
	Mitsubishi
	Nissan
	Oldsmobile

P - Z	Plymouth
	Pontiac
	Saab
	Saturn
	Subaru
	Suzuki
	Toyota
	Volkswagen
	Volvo

Figure 3.41 — Résultat de l'exemple 3.35, fichier REPORT.PDF

La mise en forme

Comme dans les procédures PRINT et TABULATE, on peut indiquer comment une colonne doit être mise en forme à l'aide d'options `STYLE` à insérer dans les instructions `PROC REPORT`, `DEFINE`, `BREAK` et `RBREAK` et `COMPUTE`. On y retrouve la syntaxe déjà vue : on indique l'emplacement mis en forme, et ensuite une série d'attributs de styles et leurs valeurs (cf. tableau 3.1).

```
STYLE(zone)=[attribut1=valeur1 < attribut2=valeur2 < … > > ]
```

Les zones mises en forme sont à moduler en fonction de l'instruction dans laquelle on insère l'option `STYLE`. Le tableau 3.2 résume quels emplacements peuvent être cités selon l'instruction.

Tableau 3.2 — Zones mises en forme selon l'instruction de la procédure REPORT

Dans l'instruction…	Zone	Où est-ce que ça se trouve ?
PROC REPORT	REPORT	Tout le rapport
	COLUMN	Tout le contenu des colonnes
	HEADER	Les cellules d'en-tête
	SUMMARY	Les récapitulatifs
	LINES	Les textes sur mesure édités par l'instruction LINE
BREAK ou RBREAK	SUMMARY	Les récapitulatifs
	LINES	Les textes sur mesure édités par l'instruction LINE
COMPUTE	LINES	Les récapitulatifs
DEFINE	COLUMN	Tout le contenu des colonnes
	HEADER	Les cellules d'en-tête

Quant aux valeurs des attributs, elles peuvent être données de manière « fixe » comme dans l'exemple 3.36, pour mettre systématiquement une colonne en valeur par exemple, ou sous conditions à l'aide d'un format, comme on l'a déjà vu pour les procédures PRINT et TABULATE, et comme l'illustre l'exemple 3.37.

Exemple 3.36 – Procédure REPORT : option STYLE

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS constructeur modele prix_max passagers ventes ;
  DEFINE constructeur / DISPLAY ;
  DEFINE modele / DISPLAY ;
  DEFINE prix_max / DISPLAY FORMAT=NLNUM12. ;
  DEFINE passagers / DISPLAY ;
  DEFINE ventes / DISPLAY STYLE(COLUMN)=[BACKGROUND=YELLOW] FORMAT=NLNUM12.;
RUN ;
```

Constructeur du véhicule	Nom du modèle	Prix du haut de gamme (FF)	Nombre de passagers	ventes
Acura	Integra	112 800	5	576
Acura	Legend	232 200	5	263
Audi	90	193 800	5	409
Audi	100	267 600	6	90
BMW	535i	217 200	4	311
Buick	LeSabre	130 200	6	2 058
Buick	Roadmaster	149 400	6	1 777

Figure 3.42 — Résultat de l'exemple 3.36 au format PDF

Exemple 3.37 – Procédure REPORT : option STYLE et formats

```
PROC FORMAT ;
 VALUE ventes
   LOW  -  500 = "GRAY70" 500 <- 1000 = "GRAYA0" 1000 <- HIGH = "GRAYD0" ;
RUN ;
PROC REPORT DATA = livre.voitures NOWD ;
 COLUMNS constructeur modele prix_max passagers ventes ;
 DEFINE constructeur / DISPLAY ;
 DEFINE modele / DISPLAY ;
 DEFINE prix_max / DISPLAY FORMAT=NLNUM12. ;
 DEFINE passagers / DISPLAY ;
 DEFINE ventes / DISPLAY FORMAT=NLNUM12. STYLE(COLUMN)=[BACKGROUND=ventes.];
RUN ;
```

Constructeur du véhicule	Nom du modèle	Prix du haut de gamme (FF)	Nombre de passagers	ventes
Acura	Integra	112 800	5	576
Acura	Legend	232 200	5	263
Audi	90	193 800	5	409
Audi	100	267 600	6	90
BMW	535i	217 200	4	311
Buick	LeSabre	130 200	6	2 058
Buick	Roadmaster	149 400	6	1 777

Figure 3.43 — Résultat de l'exemple 3.37 au format PDF

On peut également définir les attributs de style dans un bloc `COMPUTE`, ce qui permet de proposer des conditions plus complexes que via un simple format : l'exemple 3.38 pose des conditions sur une variable temporaire (permettant de colorier une ligne sur deux), tandis que l'exemple 3.39 pose des conditions sur deux variables à la fois.

```
COMPUTE … ;
   …
   CALL DEFINE (_COL_|_ROW_|"variable"|"_Cx_", "STYLE",
                "STYLE=[attribut1 valeur1 < attribut2=valeur2 < … > > ] ) ;
   …
ENDCOMP ;
```

L'emplacement précisé en premier argument de `CALL DEFINE` peut être l'intégralité de la ligne (`_ROW_`), la valeur de la cellule courante (`_COL_`), la valeur d'une autre cellule repérée par le nom de la variable ou de l'alias associé, ou encore par son numéro de colonne (en comptant, comme toujours, les colonnes non affichées par `DEFINE NOPRINT`).

Exemple 3.38 – Procédure REPORT : CALL DEFINE STYLE

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS constructeur modele prix_max passagers ventes ;
  DEFINE constructeur / DISPLAY ;
  DEFINE modele / DISPLAY ;
  DEFINE prix_max / DISPLAY ;
  DEFINE passagers / DISPLAY ;
  DEFINE ventes / DISPLAY ;
  COMPUTE BEFORE ;
    numLigne = 0 ;
  ENDCOMP ;
  COMPUTE constructeur ;
    numLigne + 1 ;
    IF MOD(numLigne,2)=0 THEN
```

```
      CALL DEFINE(_ROW_,"STYLE",
                  "STYLE=[BACKGROUND=GRAY FOREGROUND=WHITE]") ;
  ENDCOMP ;
RUN ;
```

Constructeur du véhicule	Nom du modèle	Prix du haut de gamme (FF)	Nombre de passagers	ventes
Acura	Integra	112800	5	576
Acura	Legend	232200	5	263
Audi	90	193800	5	409
Audi	100	267600	6	90
BMW	535i	217200	4	311
Buick	LeSabre	130200	6	2058
Buick	Roadmaster	149400	6	1777
Buick	Century	103800	6	1072

Figure 3.44 — Résultat de l'exemple 3.38 au format PDF

Exemple 3.39 – Procédure REPORT : CALL DEFINE STYLE et conditions

```
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS constructeur modele transmission poids ;
  DEFINE constructeur / DISPLAY ;
  DEFINE modele / DISPLAY ;
  DEFINE transmission / DISPLAY ;
  DEFINE poids / DISPLAY "Poids (kg)" ;
  COMPUTE poids ;
    poids = poids * 1000 ;
    SELECT ;
      WHEN (poids > 1500 AND transmission = "4x4") DO ;
        CALL DEFINE("poids","STYLE","STYLE=[BACKGROUND=RED]") ;
        CALL DEFINE("transmission","STYLE","STYLE=[BACKGROUND=RED]") ;
      END ;
      WHEN (poids > 1800) DO ;
        CALL DEFINE("poids","STYLE","STYLE=[BACKGROUND=RED]") ;
        CALL DEFINE("transmission","STYLE","STYLE=[BACKGROUND=RED]") ;
      END ;
      OTHERWISE ;
    END ;
  ENDCOMP ;
RUN ;
```

Constructeur du véhicule	Nom du modèle	Mode de transmission	Poids (kg)
Acura	Integra	traction	1225
Acura	Legend	traction	1613
Audi	90	traction	1529
Audi	100	traction	1542
BMW	535i	propulsion	1649
Buick	LeSabre	traction	1572
Buick	Roadmaster	propulsion	1860
Buick	Century	traction	1305
Buick	Riviera	traction	1583
Cadillac	DeVille	traction	1640
Cadillac	Seville	traction	1783
Chevrolet	Caprice	propulsion	1771
Chevrolet	Cavalier	traction	1128
Chevrolet	Corsica	traction	1262
Chevrolet	Lumina	traction	1447
Chevrolet	Lumina_APV	traction	1683
Chevrolet	Astro	4x4	1823
Chevrolet	Camaro	propulsion	1468

Figure 3.45 — Résultat de l'exemple 3.39 au format PDF

4

Les graphiques

Objectifs

Ce chapitre est consacré à la production de graphiques divers et variés avec SAS. En présence du module SAS/GRAPH, les capacités du logiciel sont assez grandes en terme de souplesse et de qualité du rendu… à condition de prendre le temps de régler les diverses options et instructions périphériques.

Nous verrons ici comment produire : des boîtes à moustaches ou *boxplots* ; des graphiques décrivant la distribution d'une variable (histogrammes et QQ-Plots) ; des diagrammes circulaires et des diagrammes en bâtons ; des nuages de points et des courbes ; des éléments superposés au graphique de votre choix.

SAS produit normalement un graphique à la fois ; la procédure GREPLAY et la notion de *template graphique* permettent de fusionner plusieurs graphiques en une seule image. La seule alternative pour présenter plusieurs graphiques ensemble est une part expérimentale du langage SAS, le GTL (lié à ODS GRAPHICS présenté à la fin de ce chapitre), qui ne sera décrit qu'au chapitre 8.

Tous ces graphiques sont produits dans SAS dans divers formats (GIF, JPEG, etc.) à travers des drivers graphiques. C'est par cette notion essentielle, et par les autres options globales affectant tous les graphiques, que commence ce chapitre.

4.1 LE FORMAT DE L'IMAGE CRÉÉE ET AUTRES OPTIONS GRAPHIQUES

Avant de produire le moindre graphique dans SAS, avec l'intention de le diffuser par la suite, il est important de fixer son choix sur le format d'image rendu. Le mode d'encodage du graphique est appelé *driver*. On distingue trois familles de drivers disponibles dans SAS :

- les formats classiques comme Gif, Jpeg (des images compressées), WMF et EMF (des images/dessins dont on peut modifier les éléments après coup) ;
- les formats actifs (des applications qui affichent une image et offrent un menu contextuel pour l'adapter a posteriori aux exigences du moment) ;
- les images fixes dérivées des formats actifs (permettant une diffusion aisée sans sacrifier l'esthétique).

Le positionnement du driver se fait à travers l'option graphique `DEVICE`. Cette option graphique est définie pour toute la durée de la session SAS, ou jusqu'à une nouvelle définition. On peut la remettre à sa valeur par défaut avec la syntaxe `GOPTION RESET=ALL ;`

```
GOPTION DEVICE = driverGraphique KEYMAP = WINANSI|ISO8859 ;
…
GOPTION RESET = ALL ; /* pour la réinitialisation des options */
```

À l'ouverture d'une session SAS la macro-variable SYSDEVIC est vide. Après exécution de `GOPTION DEVICE = driver`, la macro-variable SYSDEVIC contient le nom du driver.

4.1.1 Les drivers de base : GIF, JPEG, WMF, EMF

Les quatre premiers drivers proposés correspondent à des formats d'image standard. Les formats Gif et Jpeg sont communs, par exemple dans les pages web. Les formats WMF et EMF sont des formats de sauvegarde couramment proposés pour les dessins créés dans les applications bureautiques. Des deux derniers, EMF est le format le plus récent. Les noms de drivers sont respectivement `GIF`, `JPG`, `SASWMF` et `SASEMF`.

Dans ces formats, les graphiques sont entièrement produits par SAS ; il est donc nécessaire d'opérer quelques ajustements supplémentaires, en particulier la gestion des caractères accentués. Celle-ci se fait en deux temps : d'une part le jeu de caractères incluant les caractères accentués (option graphique `KEYMAP`, valant `WINANSI` sous Windows et `ISO8859` sur Unix) ; d'autre part l'usage par défaut d'une police contenant des caractères accentués (ce n'est pas le cas de la police par défaut). La police `SWISS`, à indiquer dans l'option `FTEXT`, est une des rares polices fournies par SAS proposant à la fois une esthétique professionnelle et des lettres accentuées.

```
GOPTION DEVICE = GIF|JPG|SASWMF|SASEMF
        KEYMAP = WINANSI|ISO8859
         FTEXT = SWISS ;
```

```
…
GOPTION RESET = ALL ; /* pour la réinitialisation des options */
```

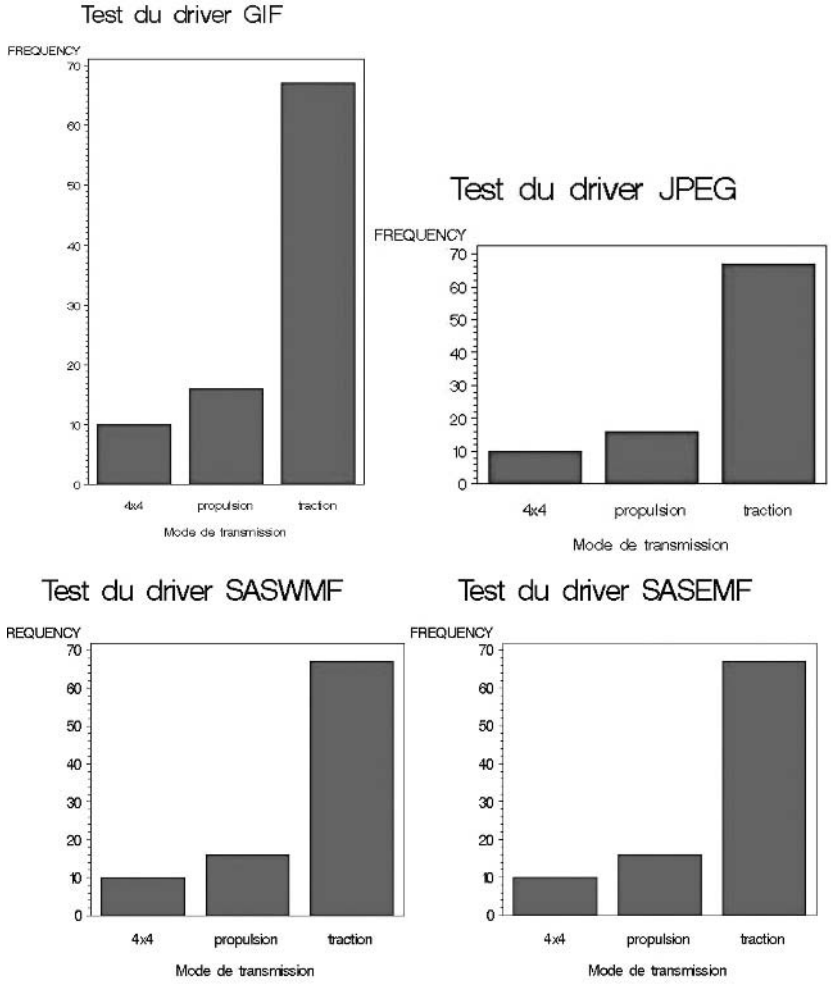


Figure 4.1 — Un histogramme produit avec différents drivers

Sur la figure 4.1, on voit que le choix du driver transforme de manière notable l'aspect du graphique final. Seuls les formats WMF et EMF sont retouchables dans Word ou PowerPoint après coup (faire un clic droit puis « Modifier l'image »).

4.1.2 Les drivers « actifs » : ActiveX, Java

Quatre autres drivers sont particulièrement intéressants : ils sont construits autour d'éléments logiciels (Java et ActiveX) qui rendent les graphiques de manière interactive. Ils ont été introduits dans la version 8 pour deux d'entre eux (JAVA et

ACTIVEX) afin d'apporter de la souplesse dans le rendu, une fois le document qui les contient produit par SAS : on peut en effet retoucher *a posteriori* le graphique grâce à un menu contextuel (clic droit).

Leur esthétique beaucoup plus moderne (pour ActiveX : voir figure 4.2 a ; pour Java, figure 4.2 b) que les drivers précédents les a rendus populaires, mais leur emploi n'est pas aisé : ils nécessitent tous les deux que la personne qui visualise le document ait l'élément logiciel en question installé sur son ordinateur. De plus, le composant ActiveX ne fonctionne *que sous Windows*. Le composant Java, lui, est disponible sur tous les systèmes d'exploitation.

Deux autres drivers ont donc été introduits avec la version 9 de SAS, appelés ACTXIMG et JAVAIMG. Ce sont des « captures d'écran » (des images figées, au format PNG) du rendu des drivers ACTIVEX et JAVA. Cette fois, pas besoin d'installer quoi que ce soit pour visualiser ces images : on s'affranchit totalement de l'environnement logiciel des destinataires du document. En revanche, on perd le menu contextuel associé aux drivers réellement actifs.

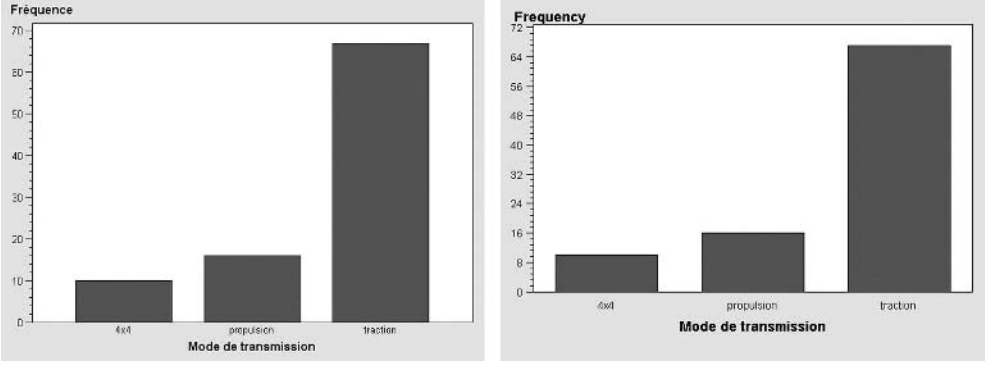


Figure 4.2 — Un histogramme produit avec les drivers : a) ACTIVEX / ACTXIMG ; b) JAVA / JAVAIMG

4.1.3 Les autres options graphiques

Il en existe de nombreuses autres options graphiques, parmi lesquelles :

- `HSIZE` et `VSIZE` qui indiquent les dimensions du graphique produit (par exemple `GOPTION HSIZE=10CM ;`) ;
- `XPIXELS` et `YPIXELS` qui indiquent la résolution (nombre de pixels en largeur et en hauteur) de l'image produite, indépendamment de sa taille ;
- `CBACK` qui gère la couleur de fond du graphique, tandis que `COLORS` énumère les couleurs à utiliser dans le graphique ;
- `FTEXT` qui mentionne la police à utiliser pour tous les textes du graphique, hormis les titres, qui sont conditionnés par l'option `FTITLE`.

4.2 LES BOÎTES À MOUSTACHES (*BOXPLOTS*)

La procédure BOXPLOT du module SAS/STAT est dédiée à la production de graphiques appelés boîtes à moustaches, boîtes à pattes ou, en anglais, *boxplots*. Extrêmement synthétiques, ces graphiques permettent de visualiser ensemble le minimum, le maximum, les quartiles et la moyenne d'une distribution.

La boxplot se présente sous la forme d'un rectangle délimité par les premier et troisième quartiles de la distribution. Ce rectangle porte une division horizontale à hauteur de la médiane, et une croix représente la moyenne. Des « pattes » ou barres verticales sont dessinées au-dessus et en dessous du rectangle jusqu'aux extrema (minimum et maximum), ou jusqu'à une distance de 1,5 fois l'écart entre les 3e et 1er quartiles.

À noter que les graphiques produits par cette procédure ne sont pas transcrits par les drivers ActiveX ou Java : on n'aura donc le choix qu'entre les drivers « passifs » décrits à la section 4.1.1.

4.2.1 Des *boxplots* côte à côte

Par défaut, la procédure BOXPLOT construit des séries de boxplots situées côte à côte en fonction d'une variable qualitative de groupe. Les données doivent impérativement être triées selon cette variable de groupe.

```
PROC SORT DATA = tableSAS OUT = tableSAStriée;
   BY < variableBlocs > variableGroupe ;
RUN ;
PROC BOXPLOT DATA = tableSAStriée ;
   PLOT variableQuanti * variableGroupe < / options > ;
 < INSET mot(s)clé(s)statistique(s) ; >
 < ID variableIdentifiant ; >
 < BY variableBlocs ; >
RUN ;
```

On indiquera dans l'instruction `PLOT` la variable quantitative qui est résumée par les boxplots ; une série d'options peut s'inscrire dans cette instruction pour améliorer l'aspect du résultat. On décrira l'option `BOXSTYLE` à la section 4.2.3. Les autres options permettent de modifier l'aspect du graphique : couleurs, labels, symboles utilisés. L'instruction `ID` permet de repérer les observations extrêmes par les valeurs d'une variable : on verra son application à la section 4.2.3 également. L'instruction `BY` permet de construire en une seule exécution de la procédure une série de graphiques, en fonction des valeurs d'une variable selon laquelle la table aura été triée au préalable.

L'instruction `INSET` permet d'incruster dans les graphiques un encadré contenant des statistiques ; on énumère derrière `INSET` un ou plusieurs mots-clés parmi les suivants : `MEAN` (affichera la moyenne), `MIN` et `MAX` (les extrema sur l'ensemble des données), `NMIN` et `NMAX` (les effectifs du plus petit et du plus grand groupe), `STDDEV` (la

racine carrée de la somme pondérée des variances intra-groupes – une sorte d'écart-type intra-groupes).

Exemple 4.01 – Boxplots côte à côte

```
PROC SORT DATA = livre.voitures
          OUT = work.voitures ;
   BY type ;
RUN ;
GOPTION KEYMAP = WINANSI ;
PROC BOXPLOT DATA = work.voitures ;
   PLOT conso_auto * type ;
   INSET MEAN NMIN NMAX NOBS ;
   LABEL conso_auto = "Consommation sur autoroute" ;
RUN ;
```

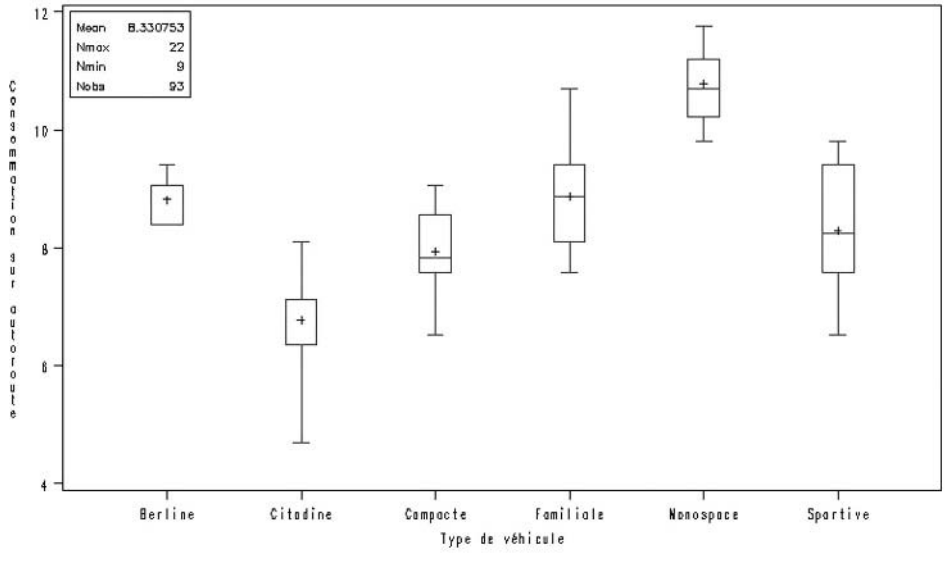


Figure 4.3 — Résultat de l'exemple 4.01 (driver GIF)

4.2.2 Une seule *boxplot*

La procédure BOXPLOT n'accepte de produire ses graphiques que si on indique deux variables dans `PLOT`. On doit ruser en construisant une variable constante qui servira à grouper, pour obtenir un graphique ne contenant qu'une seule boxplot.

Exemple 4.02 – Boxplot seule

```
DATA work.voitures ;
       SET livre.voitures ;
       constante = "Prix" ;
RUN ;
GOPTION KEYMAP = WINANSI ;
```

```
PROC BOXPLOT DATA = work.voitures ;
       PLOT prix_max * constante ;
       LABEL constante = "du haut de gamme"
          prix_max = "francs 1993" ;
RUN ;
```

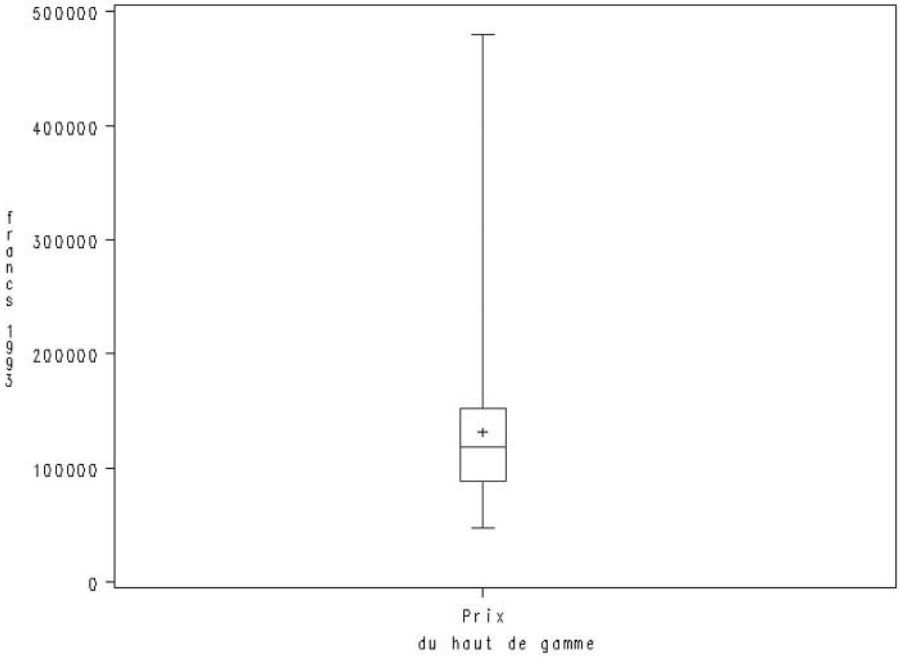


Figure 4.4 — Résultat de l'exemple 4.02 (driver GIF)

4.2.3 Les individus extrêmes

La procédure BOXPLOT propose plusieurs types d'affichage des pattes *via* une option BOXSTYLE.

```
PROC BOXPLOT DATA = tableSAS ;
   PLOT variableQuanti * variableGroupe / BOXSTYLE = SCHEMATICID ;
   ID variableIdentifiant ;
RUN ;
```

- par défaut (BOXSTYLE=SKELETAL), les pattes vont jusqu'au minimum et au maximum de la distribution ;
- l'option BOXSTYLE=SCHEMATIC fait que les pattes vont du premier ou du troisième quartile jusqu'à 1,5 fois l'écart entre ces deux quartiles. Les valeurs qui se trouveraient au-delà de la longueur des pattes sont repérées par des points isolés ;

- l'option `BOXSTYLE=SCHEMATICID` permet d'avoir, en plus du cas précédent, un affichage des points extrêmes accompagné de la valeur d'un identifiant. La variable contenant cet identifiant est indiquée dans l'instruction `ID`.

Exemple 4.03 – Identification des individus extrêmes dans une boxplot

```
PROC SORT DATA = livre.voitures OUT = work.voitures ;
       BY type ;
RUN ;
PROC BOXPLOT DATA = work.voitures ;
       PLOT poids * type / BOXSTYLE = SCHEMATICID ;
       ID modele ;
RUN ;
```

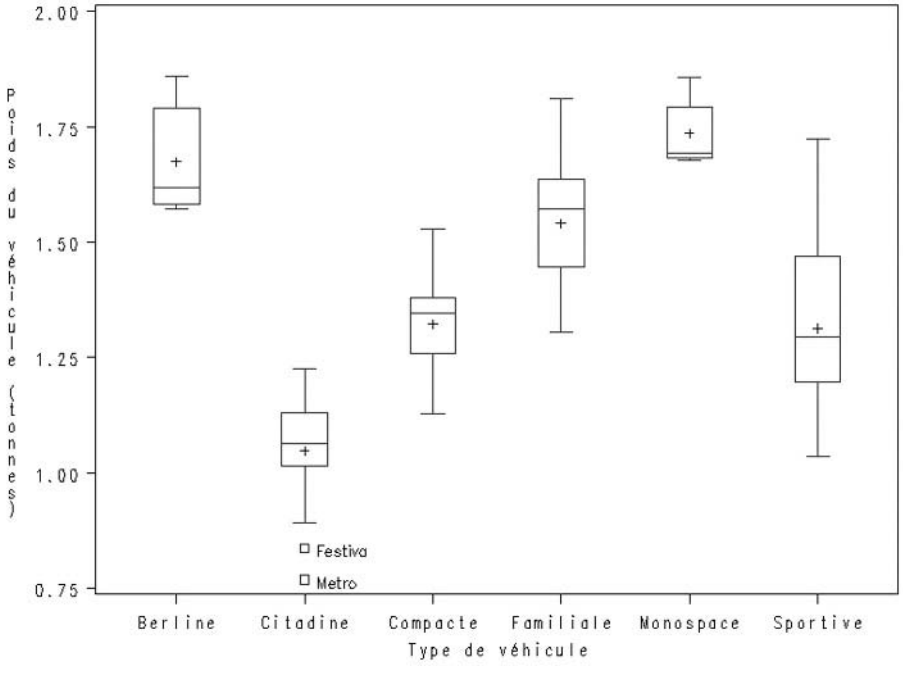


Figure 4.5 — Résultat de l'exemple 4.03 (driver GIF)

4.3 LA DISTRIBUTION D'UNE VARIABLE (PROCÉDURE UNIVARIATE)

La procédure UNIVARIATE n'a longtemps été qu'une sorte de super-procédure MEANS, apte à produire de nombreuses statistiques (quantiles, tests de normalité, mode) mais dont les graphiques (*boxplot*, *stem and leaf*, *QQ-plot*) étaient d'une extrême indigence. Depuis la version 8, en présence d'une licence pour SAS/

GRAPH, des histogrammes et des graphiques d'ajustement à une loi (QQ-plots) sont aisément produits, avec une esthétique très satisfaisante.

À partir de la version 9.2, ces graphiques sont produits automatiquement *via* le système ODS GRAPHICS (voir la section 4.9 et chapitre 8), ce qui les rend encore plus beaux.

4.3.1 Les histogrammes

L'instruction `HISTOGRAM` permet de demander la création d'un graphique en bâtons résumant la distribution d'une variable quantitative. Le nombre de bâtons est automatiquement calculé par SAS, même si on dispose d'options pour déterminer les valeurs sur lesquelles les barres sont centrées. On peut profiter des options de cette instruction pour réclamer la superposition sur cet histogramme de la courbe de densité d'une ou de plusieurs lois connues (sont disponibles : normale, log-normale, Gamma, Bêta, Weibull, exponentielle). Les paramètres (moyenne, variance, etc.) de ces lois sont soit connus de l'utilisateur, soit estimés directement dans les données.

On peut faire suivre cette instruction `HISTOGRAM` d'une instruction `INSET` qui permet d'incruster une série de statistiques : moyenne, médiane, nombre d'observations, variance, et même les statistiques des tests d'adéquation à des lois (tests de Kolmogorov-Smirnov [`KSD`], Cramer-Von Mises [`CVM`] et Anderson-Darling [`AD`]).

```
PROC UNIVARIATE DATA = tableSAS ;
   VAR variable(s)Quantitative(s) ;
 < CLASS variableGroupe ; >
   HISTOGRAM variable(s)Quantitative(s) < / loi(COLOR=couleur
                                              L=courbe W=épaisseur) > ;
 < INSET statistique1="intitulé" < statistique2="intitulé" > < … > ; >
RUN ;
```

Dans l'instruction `HISTOGRAM`, on demande l'ajout de la densité d'une loi par l'ajout dans les options du ou des mots-clés `NORMAL`, `LOGNORMAL`, `GAMMA`, `BETA`, `WEIBULL` ou `EXPONENTIAL`. À côté de chacun de ces mots-clés, on peut préciser entre parenthèses `COLOR=` suivi de la couleur souhaitée pour cette courbe (la liste des couleurs est celle indiquée dans la section 2.2.1), `L=` suivi du type de courbe (1 pour un trait plein, 2 à 46 pour des pointillés) ou encore `W=` suivi de l'épaisseur du trait (1 par défaut, mais on peut proposer un nombre entier dont l'esthétique et la lisibilité demanderont qu'il soit inférieur à 6).

Les statistiques que l'on peut réclamer dans l'instruction `INSET` sont nombreuses : les principales correspondent aux mots-clés que l'on retrouve dans la procédure MEANS et dans l'instruction `OUTPUT` de la procédure UNIVARIATE. Il s'agit de `MEAN`, `STD`, `MIN`, `MAX`, `NOBS`, `NMISS`, `CV` (coefficient de variation), des quantiles usuels (`P1`, `P5`, `P10`, `Q1`, `MEDIAN`, `Q3`, `P90`, `P95`, `P99`), de l'écart interquartile `QRANGE` (il vaut Q3-Q1), de la valeur la plus fréquente (`MODE`). On retrouve encore les statistiques de test et p-values associées pour les tests d'adéquation à une des susdites lois : la forme générale est `nomLoi(nomTest)` ou `nomLoi(nomTestPVAL)`, pour obtenir respectivement

la statistique de test ou la p-value du test. On écrira par exemple `NORMAL(KSDPVAL)` pour la p-value du test de normalité de Kolmogorov-Smirnov.

Exemple 4.04 – Histogramme avec superposition de lois log-normale et Gamma

```
GOPTION KEYMAP = WINANSI ;
PROC UNIVARIATE DATA = livre.voitures ;
   VAR puissance ;
   HISTOGRAM puissance / GAMMA(COLOR=BLACK L=1) LOGNORMAL(COLOR=BLACK L=3) ;
   INSET MEAN MEDIAN GAMMA(KSDPVAL) LOGNORMAL(KSDPVAL) ;
RUN ;
```

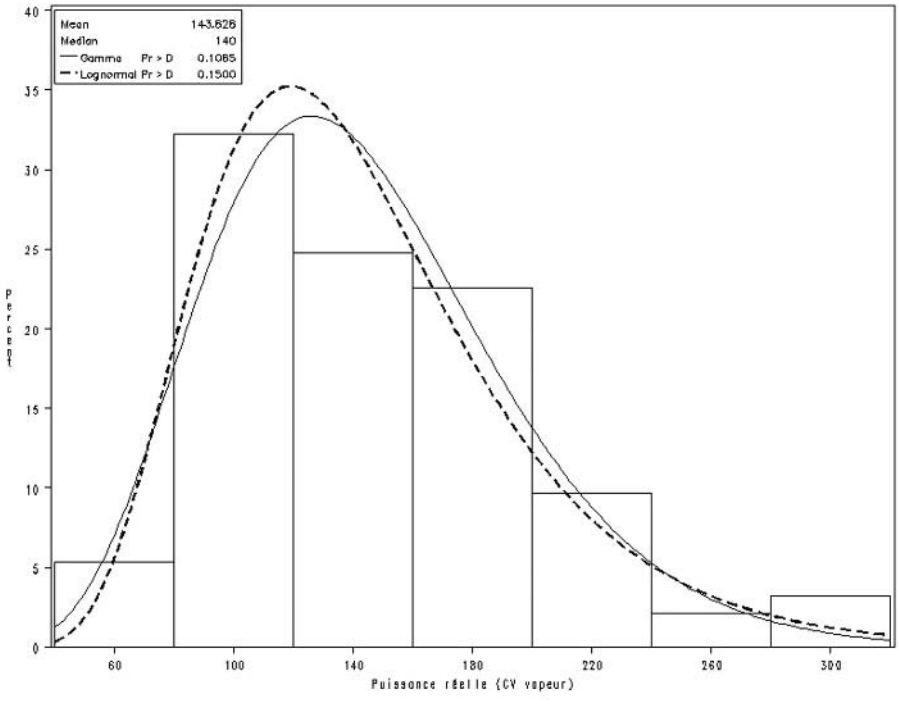


Figure 4.6 — Résultat de l'exemple 4.04 (driver GIF)

Les probabilités des tests conforment l'impression visuelle : la courbe suivant une loi log-normale (celle en pointillés) est celle qui s'ajuste le mieux à la répartition de la puissance parmi les véhicules étudiés.

4.3.2 Les graphiques d'adéquation à une loi (*QQ-Plots*)

Le *QQ-plot* est un graphique portant en abscisse les quantiles théoriques, et en ordonnée les quantiles observés. Une droite barre ce graphique : si la variable suit

d'assez près la loi théorique, les points correspondant aux quantiles calculés doivent être près de cette droite.

On requiert ce graphique à l'aide de l'instruction QQPLOT ; on y indique la variable étudiée, ainsi que la loi à laquelle on souhaite la comparer. Contrairement à l'instruction HISTOGRAM, il est ici obligatoire d'évoquer pour chaque loi ses paramètres : on leur donnera soit des valeurs, soit le mot-clé EST viendra indiquer qu'ils sont à estimer empiriquement à partir des données. L'option SQUARE permet d'obtenir un graphique carré, plus aisé à lire.

```
PROC UNIVARIATE DATA = tableSAS ;
   VAR variable ;
   QQPLOT variable / SQUARE loi (paramètre1=EST|valeur
                                 < paramètre2=EST|valeur > < … >) ;
< INSET statistique1="intitulé" < statistique2="intitulé" > < … > ; >
RUN ;
```

Tableau 4.1 — Paramètres associés aux lois disponibles dans la procédure UNIVARIATE

Loi (mot-clé)	Paramètres
NORMAL	MU, SIGMA
LOGNORMAL	SIGMA, THETA, ZETA ou SLOPE
GAMMA	ALPHA, SIGMA, THETA
EXPONENTIAL	SIGMA, THETA
BETA	ALPHA, BETA, SIGMA, THETA
WEIBULL	C, SIGMA, THETA

On retrouve ici l'instruction INSET permettant d'afficher dans un encadré les statistiques de la variable étudiée. À noter qu'on ne peut pas insérer dans un histogramme les statistiques de test (KSD, AD et CVM) ni leurs p-values. En revanche, quantiles, moyenne, écart-type et effectif sont régis par la même syntaxe qu'à la section précédente.

Exemple 4.05 – QQ-plot d'adéquation à une loi log-normale

```
PROC UNIVARIATE DATA = livre.voitures ;
       VAR puissance ;
       QQPLOT puissance / SQUARE LOGNORMAL(THETA=EST SIGMA=EST ZETA=EST) ;
       INSET MEAN="Moyenne" MEDIAN="Médiane" NOBS="Nb obs" ;
RUN ;
```

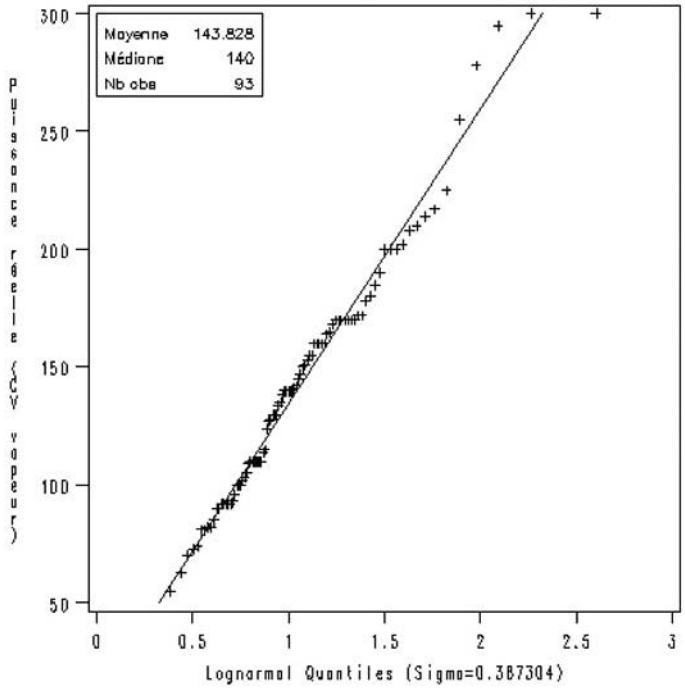


Figure 4.7 — Résultat de l'exemple 4.05 (driver GIF)

4.4 LES DIAGRAMMES CIRCULAIRES ET EN BÂTONS

Cette partie constitue un développement des bases à connaître sur le fonctionnement de la procédure GCHART. Après un rapide rappel des options les plus courantes[1], on s'intéressera aux astuces permettant de représenter deux quantités en miroir sur un même diagramme, comme dans les pyramides des âges, puis au cas des diagrammes circulaires, qui récompensent le choix judicieux de quelques options par une impression visuelle flatteuse.

4.4.1 Les histogrammes et autres diagrammes en bâtons

La procédure GCHART propose quatre instructions pour éditer des diagrammes en bâtons (appelés histogrammes dans le cas particulier où ils résument la distribution d'une donnée continue) : `VBAR` et `VBAR3D` pour les bâtons verticaux et `HBAR` ou `HBAR3D` pour les bâtons horizontaux. Dans ce dernier type de graphique, des statisti-

1. Pour de plus amples informations sur ces options, on se référera par exemple à *SAS – Maîtriser SAS Base et SAS Macro, 2e édition*, H. Kontchou et O. Decourt, Dunod, chapitre 4.

ques sont automatiquement éditées à la droite des bâtons (on ne les affiche pas grâce à l'option NOSTATS).

```
PROC GCHART DATA = tableSAS ;
   VBAR|VBAR3D variable < / DISCRETE DESCENDING
                            TYPE=FREQ|PERCENT|CFREQ|CPERCENT|SUM|MEAN
                            SUMVAR=varNumérique
                            GROUP=varGpe1 SUBGROUP=varGpe2 > ;
   HBAR|HBAR3D variable < / DISCRETE DESCENDING NOSTATS
                            TYPE=FREQ|PERCENT|CFREQ|CPERCENT|SUM|MEAN
                            SUMVAR=varNumérique
                            GROUP=varGpe1 SUBGROUP=varGpe2 > ;
RUN ; QUIT ;
```

Dans le cas où la variable représentée est numérique, on a le choix d'omettre l'option DISCRETE pour obtenir un histogramme, ou de l'activer pour que chaque valeur de la variable se traduise par un bâton. L'option DESCENDING permet, elle, d'ordonner les bâtons par taille décroissante (par défaut, ce sera selon la variable représentée).

L'option TYPE contrôle la dimension des bâtons : proportionnelle par défaut au nombre d'observations (TYPE=FREQ), elle peut aussi être fonction du pourcentage que ce nombre représente (TYPE=PERCENT), à des fréquences ou des pourcentages cumulés (TYPE=CFREQ et CPERCENT) ou encore à la moyenne ou la somme (TYPE=MEAN ou SUM) d'une variable numérique qui sera citée dans l'option SUMVAR.

Les options GROUP et SUBGROUP permettent de représenter deux autres variables (impérativement qualitatives) sur les graphiques : GROUP en juxtaposant plusieurs séries de bâtons, SUBGROUP en subdivisant les bâtons en zones de couleurs différentes.

Exemple 4.06 – Diagramme en bâtons

```
PROC GCHART DATA = livre.voitures ;
       VBAR type / GROUP=transmission
                   TYPE=MEAN SUMVAR=poids
                   DESCENDING ;
RUN ; QUIT ;
```

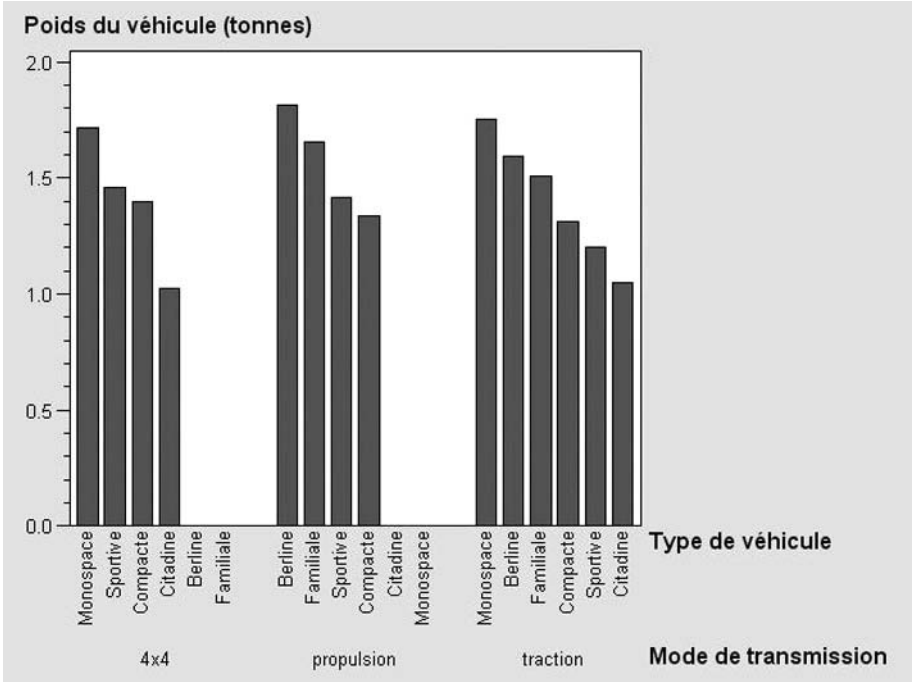


Figure 4.8 — Résultat de l'exemple 4.06 (driver ActiveX)

4.4.2 Les pyramides

Si la variable représentée *via* l'option `SUMVAR` prend des valeurs négatives, le bâton est représenté vers le bas dans un diagramme `VBAR`, et vers la gauche dans `HBAR`. On peut jouer de ce comportement pour représenter deux quantités identiques, dans deux sous-groupes qu'on ne souhaite pas voir se superposer. On obtient ainsi des graphiques de type « pyramide des âges » – c'est en effet l'application la plus courante de cette présentation des données.

Exemple 4.07 – Graphique en pyramide : répartition par puissance des véhicules

```
DATA work.voitures ;
       SET livre.voitures ;
       nb = IFN(americaine = "1", 1, -1) ;
       puissance = ROUND(puissance,10) ;
RUN ;
PROC GCHART DATA = work.voitures ;
       HBAR puissance / SUBGROUP=americaine NOSTATS TYPE=SUM SUMVAR=nb ;
RUN ; QUIT ;
```

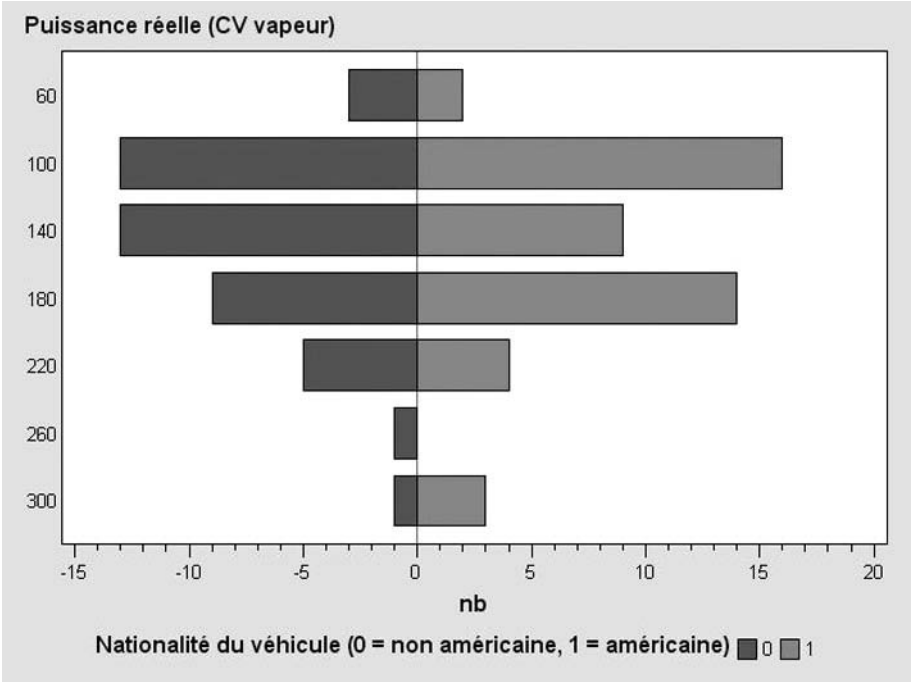


Figure 4.9 — Résultat de l'exemple 4.07 (driver ActiveX)

Dans l'exemple 4.07, on transforme les données de manière à avoir une seule variable, NB, qui contient un poids de 1 pour les voitures américaines et de -1 pour les autres, avant de représenter la somme des poids par tranche de puissance (et avec la nationalité du constructeur comme variable `SUBGROUP`). Pour produire une pyramide des âges, on procéderait à l'identique en utilisant l'âge au lieu de PUISSANCE, et une variable SEXE au lieu de la nationalité du constructeur.

4.4.3 Les diagrammes circulaires ou camemberts

La base de la production de diagrammes circulaires est l'instruction `PIE` de la procédure GCHART. Elle peut être remplacée par `PIE3D` ou `DONUT`, qui donnent respectivement un graphique avec effet de relief ou un camembert troué au centre (en forme de beignet américain !).

```
PROC GCHART DATA = tableSAS ;
   PIE|PIE3D|DONUT variable < / DISCRETE MISSING
                              OTHER=pourcentage OTHERLABEL="texte"
                              TYPE=FREQ|PERCENT|SUM SUMVAR=varNumérique
                              VALUE|SLICE|PERCENT=INSIDE|OUTSIDE|ARROW|NONE
                              EXPLODE=valeur1 < valeur2 … > >;
RUN ; QUIT ;
```

L'option DISCRETE a la même fonction que précédemment ; de même pour TYPE et SUMVAR. MISSING prend en compte la valeur manquante comme une valeur à part entière (et l'intègre donc comme une tranche du diagramme). OTHER indique au-dessous de quel pourcentage du total une tranche est fusionnée dans une catégorie « Autres » dont on indiquera le libellé dans OTHERLABEL.

Les options VALUE, SLICE et PERCENT positionnent respectivement la valeur de la statistique demandée (cf. TYPE), la valeur représentée dans ce secteur et le pourcentage associé : INSIDE dans la tranche, OUTSIDE à côté de la tranche et ARROW à l'extérieur du graphique, avec une flèche indiquant le secteur. L'option EXPLODE qui permet de « détacher » du diagramme certaines tranches que l'on veut mettre en exergue. Les valeurs correspondant aux secteurs à extraire sont citées à la suite les unes des autres ; une variable de type caractère demande qu'on indique ces valeurs entre guillemets.

Exemple 4.08 – Diagramme circulaire

```
PROC GCHART DATA=livre.voitures ;
       PIE3D type / VALUE=ARROW PERCENT=ARROW SLICE=ARROW
                    TYPE=FREQ EXPLODE="Familiale" ;
RUN ; QUIT ;
```

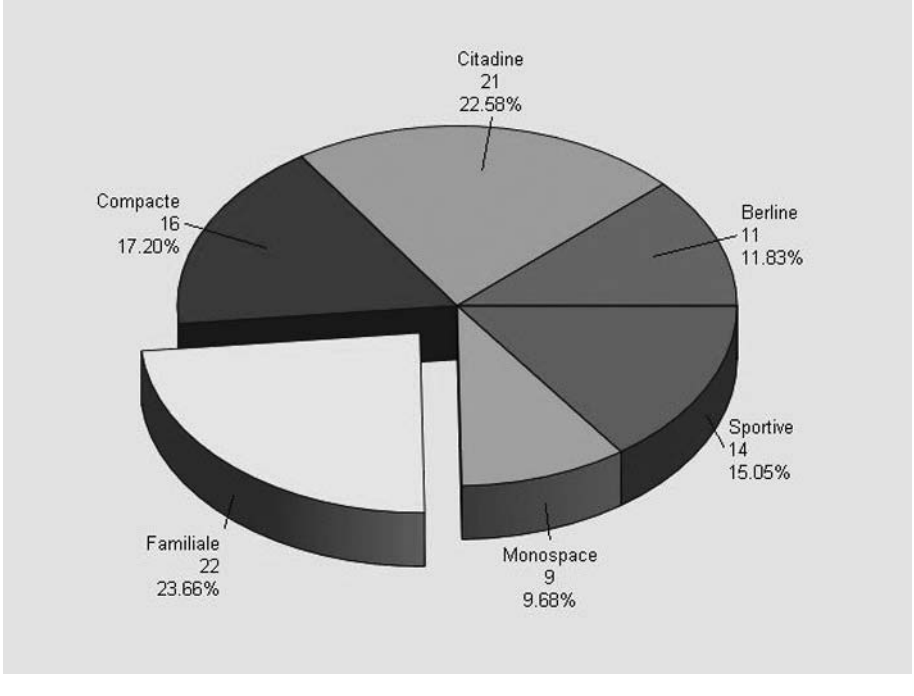


Figure 4.10 — Résultat de l'exemple 4.08 (driver ActiveX)

4.4.4 La navigation (*drill-down*) et les info-bulles

Il est possible, pour tous les types de graphiques présentés ici, de construire des zones cliquables (si le graphique est inclus dans une page HTML via l'ODS) : ce seront les barres ou les secteurs d'un diagramme circulaire. Pour cela, il faut avoir préparé au préalable une variable de type caractère contenant une partie du code HTML associé au lien (sa valeur doit être `HREF="nom fichier cible"`). On appelle cette fonctionnalité « *drill-down* » car elle permet d'aller vers plus de détails (par exemple, un lien hypertexte allant du graphique précédent à une page avec les caractéristiques de l'ensemble des véhicules de ce type).

Cette variable aura des valeurs pour toutes les observations de la table à associer à un lien. On indiquera son nom dans l'option `HTML=` de l'instruction graphique (`HBAR`, `VBAR`, `PIE`, etc.).

Il est également possible de construire par ce moyen des info-bulles s'affichant quand on laisse la souris au-dessus d'une barre ou d'un secteur angulaire ; la variable texte contiendra des valeurs comme `TITLE="texte info-bulle"`. Il est possible d'avoir à la fois des liens et des info-bulles en concaténant les valeurs `HREF="fichier"` et `TITLE="texte"`. Le texte `"0A"x` indiquera dans l'info-bulle un retour à la ligne.

Exemple 4.09 – Diagramme circulaire avec info-bulles

```
/* calcul des statistiques par type de véhicule */
ODS OUTPUT summary = work.stats ;
PROC MEANS DATA = livre.voitures MEAN ;
       VAR conso_: ;
       CLASS type ;
RUN ;
/* jointure avec les données pour le graphique */
PROC SORT DATA = livre.voitures OUT = work.voitures ;
       BY type ;
RUN ;
DATA work.voitures ;
       MERGE work.voitures
                work.stats ;
       BY type ;
       LENGTH bulle $ 200 ; /* prévoir une longueur importante */
       bulle = 'TITLE="Conso moyenne...'!!"0A"x!!
                "ville : "!!PUT(conso_ville_mean, NUMX6.1)!!" L/100 km"!!
                "0A"x!!
                "autoroute : "!!PUT(conso_auto_mean , NUMX6.1)!!' L/100 km"'
       ;
       bulle = COMPBL(bulle) ; /* élimination des blancs inutiles */
RUN ;
ODS HTML FILE="c:\temp\infobulles.htm" ;
PROC GCHART DATA=work.voitures ;
       PIE3D type / TYPE=PERCENT EXPLODE="Familiale" HTML=bulle ;
RUN ; QUIT ;
ODS HTML CLOSE ;
```

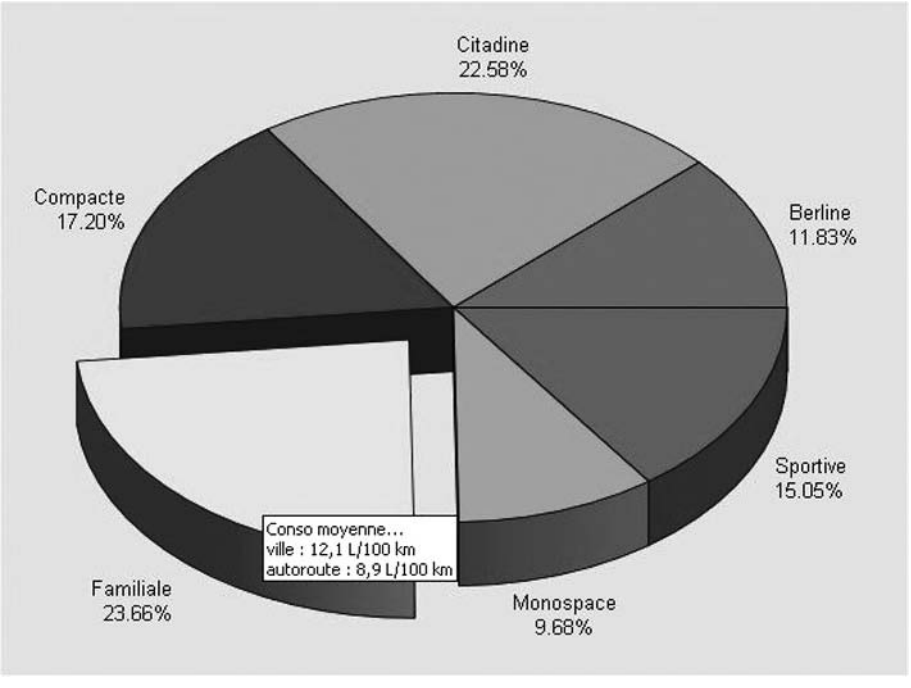


Figure 4.11 — Résultat de l'exemple 4.09 (driver ActiveX)

4.4.5 Le choix des couleurs

Quel que soit le type de graphique édité, circulaire ou en bâtons, il est possible de forcer le choix des couleurs qui y figurent, à l'aide de l'instruction `PATTERN`. Celle-ci est globale, comme un titre, et ne fait donc pas spécifiquement partie de la syntaxe de la procédure GCHART ; comme un titre, elle a un effet permanent, et doit donc être régulièrement réinitialisée.

Il est possible de définir jusqu'à 255 instructions `PATTERN`, numérotées. Elles seront utilisées selon l'ordre des valeurs associées aux différentes couleurs du graphique. Sans instruction `PATTERN`, SAS utilise les couleurs dans l'ordre où elles sont citées dans l'option graphique `COLORS` et boucle quand il a épuisé la liste (par défaut `COLORS` vaut `BLACK RED GREEN BLUE CYAN MAGENTA GRAY PINK ORANGE BROWN`).

```
PATTERN1 C = couleur1 ;
< PATTERN2 C = couleur2 ; >
PROC GCHART DATA = tableSAS ;
   … ;
RUN ; QUIT ;
GOPTION RESET = PATTERN ;
```

Les couleurs que l'on peut spécifier dans une instruction `PATTERN` sont :

- Les mots-clés correspondant à des couleurs prédéfinies : `BLACK, BLUE, BROWN, CHARCOAL, CREAM, CYAN, GOLD, GRAY, GREEN, LILAC, LIME, MAGENTA,`

MAROON, OLIVE, ORANGE, PINK, PURPLE, RED, ROSE, SALMON, STEEL, TAN, VIOLET, WHITE, YELLOW et tous les autres codes couleurs références dans le registre SAS (voir la note en bas de page dans la section 2.2 sur l'utilisation de la proc Registry pour lister ces couleurs).

- Les codes RGB (habituellement utilisés dans les feuilles de styles des pages HTML) où les 6 « chiffres » du code hexadécimal sont à faire précéder de CX (en lieu et place du # qu'on trouve en HTML). On les énonce entre guillemets : l'option COLOR="CX0000AA" écrit le titre en bleu foncé.
- Les niveaux de gris (GRAY*hh*) : les *hh* sont à remplacer par deux chiffres en code hexadécimal, de 00 (noir) à FF (blanc).

4.5 LES NUAGES DE POINTS ET LES COURBES

La procédure GPLOT permet d'obtenir des graphiques autour d'un nuage de points : tel quel, soit en connectant les points, soit encore en proposant lissages et droites de régression à travers le nuage. Après un bref rappel des options de base[1], on étudiera particulièrement les variantes proposées par l'instruction SYMBOL, les graphiques à deux échelles, les graphiques en bulles, et enfin comment, à l'aide de la procédure KDE, on peut représenter plusieurs distributions sur un seul graphique – une alternative aux histogrammes qui ne sont pas superposables.

4.5.1 Les nuages de points

La syntaxe de base s'articule autour de l'instruction PLOT, dans laquelle on indique (dans cet ordre) la variable en ordonnées puis celle en abscisses. Si on a plusieurs couples de variables à représenter, on peut user de factorisation : ainsi (varY1 varY2) * varX est identique à varY1*varX varY2*varX. Par défaut, un graphique différent est édité par couple de variables ; si on souhaite les superposer, on recourra à l'option OVERLAY.

```
PROC PLOT DATA = tableSAS ;
   PLOT varY * varX <= varGroupe > < / OVERLAY HREF=valeur VREF=valeur >;
RUN ; QUIT ;
```

Il est également possible d'obtenir des courbes superposées si elles correspondent à différents groupes d'observations : à ce moment-là, la syntaxe sera varY * varX = varGroupe et l'option OVERLAY ne sera pas nécessaire. L'emploi de l'une ou l'autre syntaxe pour superposer des courbes dépend principalement de l'organisation de la table en entrée. Les options HREF et VREF permettent d'indiquer à quel niveau on souhaite tracer des lignes de référence.

1. Ces options sont par exemple détaillées dans *SAS – Maîtriser SAS Base et SAS Macro*, op. cit.

Exemple 4.10 – Nuage de points par groupe

```
PROC GPLOT DATA = livre.voitures ;
  PLOT ventes * puissance  = americaine ;
RUN ; QUIT ;
```

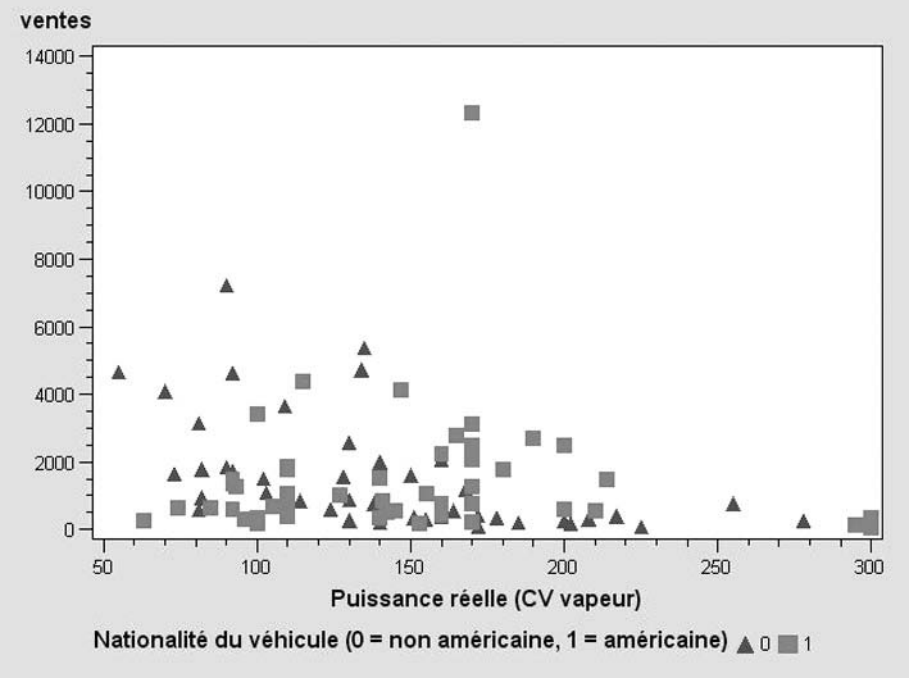


Figure 4.12 — Résultat de l'exemple 4.10 (driver ActiveX)

4.5.2 Les courbes et autres variantes

On peut obtenir des variantes de ces nuages de points en jouant sur l'instruction `SYMBOL`. On peut ainsi obtenir des courbes, lissées ou non, ou encore changer l'aspect des points.

Cette instruction étant globale comme un titre (il s'agit d'une option graphique), elle a un effet permanent et il est conseillé de la réinitialiser après usage. Dans cette instruction, on indique le marqueur des points (V= suivi d'un code parmi ceux du tableau 4.2) et le type de graphique (I= suivi du genre de courbe que l'on souhaite). Avec `I=NONE` (par défaut), on a un nuage de points. Avec `I=JOIN`, une courbe simple. Avec `I=SM`, on a une courbe lissée. Avec `I=RL`, une droite de régression est tracée, qui passe au plus près d'un maximum de points. Avec `I=RLCLM95`, on a en plus de la droite de régression un intervalle de confiance à 95 % autour de la moyenne. Avec `I=STEPJ`, on a une droite en escalier (les points sont joints par des segments horizontaux et verticaux).

L'option `W` indique la largeur du trait (1 par défaut, tout entier est possible) et `L` le tracé (1 pour ligne pleine par défaut, de 2 à 46 pour différents types de pointillés). L'option `POINTLABEL` permet d'ajouter les valeurs d'une variable à côté de chaque point. Le nom de la variable doit apparaître entre guillemets, précédé d'un dièse, le tout entre parenthèses !

```
SYMBOL I=typeCourbe V=marqueur L=trait W=largeur POINTLABEL=("#variable") ;
PROC PLOT DATA = tableSAS ;
   PLOT vary * varX <= varGroupe > < / OVERLAY HREF=valeur VREF=valeur >;
RUN ; QUIT ;
```

Il est *indispensable* que les données soient *triées* selon la variable représentée en abscisses dans la table lue par la procédure GPLOT : en effet, *les points sont reliés dans l'ordre où la procédure les lit dans la table SAS*.

Tableau 4.2 — Principaux marqueurs de points utilisables dans `SYMBOL`

Mot-clé	Forme	Mot-clé	Forme
`NONE`	Pas de marqueur	`STAR`	Astérisque ou étoile
`PLUS`	Croix droite +	`x`	Croix oblique X
`DOT`	Cercle plein	`CIRCLE`	Cercle vide
`SQUARE`	Carré	`&`	Trèfle à 3 feuilles
`DIAMOND`	Losange	`#`	Cœur
`TRIANGLE`	Triangle équilatéral	`>`	Mars (symbole masculin)
`POINT`	Point (petit)	`*`	Vénus (symbole féminin)

Exemple 4.11 – Différents types de courbes avec l'instruction `SYMBOL`

```
ODS OUTPUT summary = work.stats ;
PROC MEANS DATA = livre.voitures SUM ;
  VAR ventes ;
  CLASS puissance ;
RUN ;
SYMBOL I = JOIN V = # ;
PROC GPLOT DATA=work.stats ;
  PLOT ventes_sum * puissance ;
RUN ; QUIT ;
SYMBOL I = RLCLM95 V = STAR ;
PROC GPLOT DATA=work.stats ;
  PLOT ventes_sum * puissance ;
RUN ; QUIT ;
SYMBOL I = SM V = DOT ;
PROC GPLOT DATA=work.stats ;
  PLOT ventes_sum * puissance ;
RUN ; QUIT ;
GOPTION RESET=SYMBOL ;
```

Les figures 4.13, 4.14 et 4.15 présentent les résultats de ces différents graphiques.

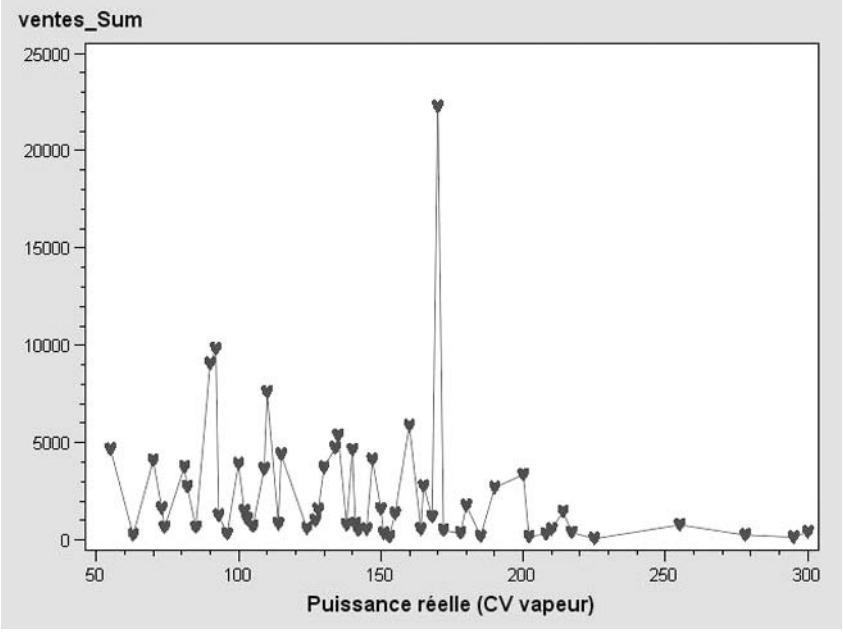


Figure 4.13 — Résultat de l'exemple 4.11 (driver ActiveX) : courbe simple

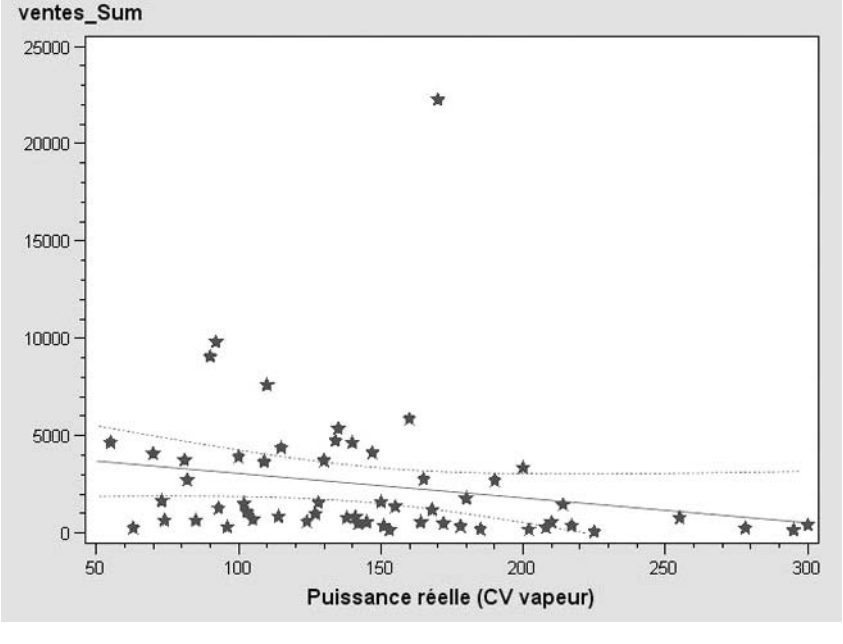


Figure 4.14 — Résultat de l'exemple 4.11 (driver ActiveX) : régression linéaire et intervalle de confiance à 95 %

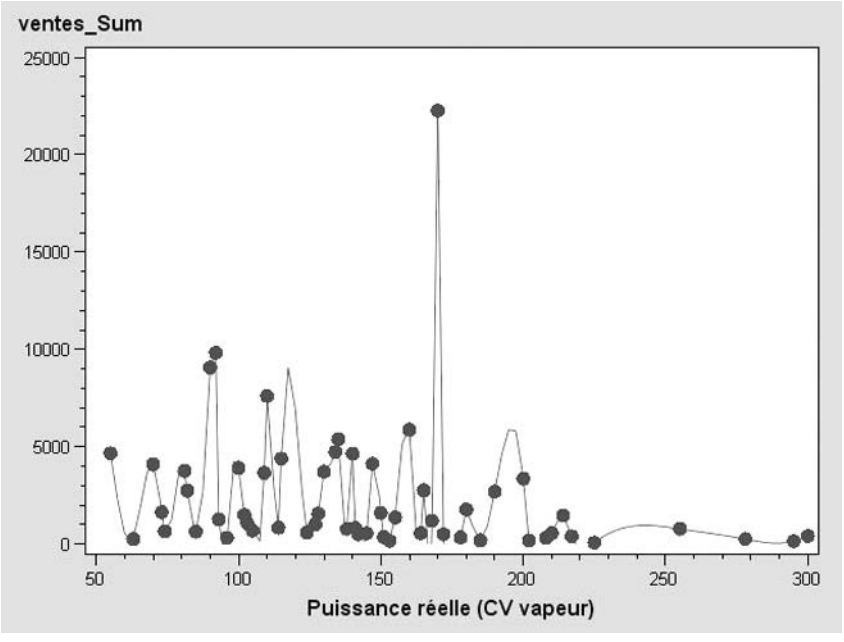


Figure 4.15 — Résultat de l'exemple 4.06 (driver ActiveX) : courbe lissée

4.5.3 Les graphiques à deux échelles

Pour représenter simultanément deux quantités d'ordres de grandeur très différents, il est possible de recourir à un graphique à deux échelles : on ajoute une instruction `PLOT2` qui reprend la même variable en abscisses que `PLOT`, mais indique une autre variable en ordonnées.

```
SYMBOL I = typeCourbe V = marqueur POINTLABEL = ("#variable") ;
PROC PLOT DATA = tableSAS ;
   PLOT vary1 * varX <= varGroupe > < / HREF=valeur VREF=valeur > ;
   PLOT2 vary2 * varX <= varGroupe > < / HREF=valeur VREF=valeur > ;
RUN ; QUIT ;
```

Hormis quand on indique une variable de groupe, il n'est pas possible de savoir à quelles quantités (et à quelles couleurs) correspondent les courbes. Il faut donc le deviner, ou arriver à générer une légende par des moyens détournés (le graphique de l'exemple 4.12 sera amélioré par une légende dans la section 4.6.1 qui traite des légendes personnalisées, illustrant un des contournements possibles des choix par défaut).

Exemple 4.12 – Graphique à double échelle

```
ODS OUTPUT summary = work.stats ;
PROC MEANS DATA = livre.voitures SUM MEAN ;
  VAR ventes poids ;
  CLASS puissance ;
```

```
RUN ;
SYMBOL i = join ;
PROC GPLOT DATA = work.stats ;
  PLOT ventes_sum * puissance ;
  PLOT2 poids_mean * puissance ;
  LABEL ventes_sum = "Ventes totales"
        poids_mean = "Poids moyen" ;
RUN ; QUIT ;
GOPTION RESET = SYMBOL ;
```

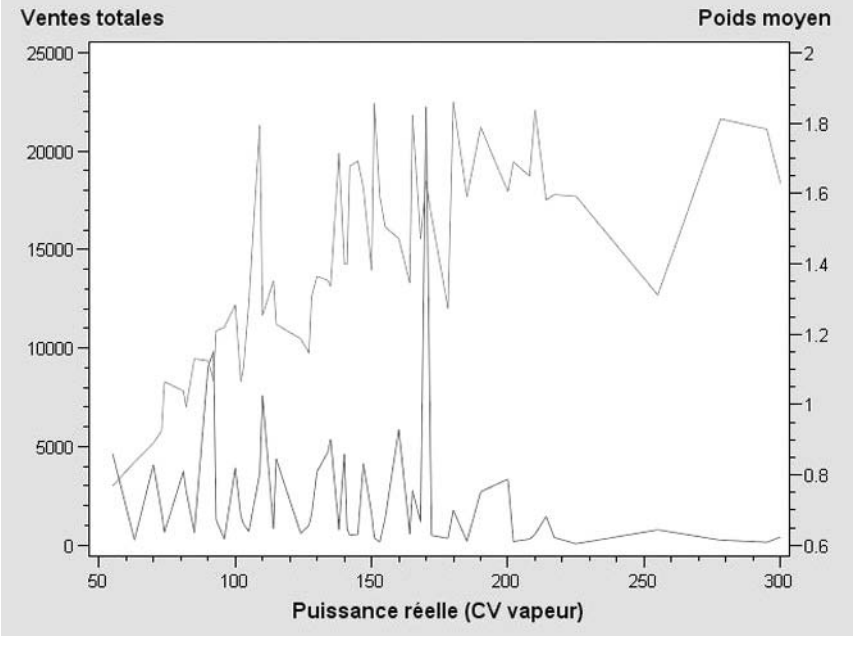


Figure 4.16 — Résultat de l'exemple 4.12 (driver ActiveX)

4.5.4 Les séries incomplètes

Par défaut, quand les données présentent une valeur manquante, SAS relie les points qui précèdent et suivent la valeur manquante sans aucune mention d'un éventuel problème. On peut cependant interrompre la série avec l'option `SKIPMISS` (admise dans les instructions `PLOT` et `PLOT2`). En sa présence, la courbe s'arrête à l'observation précédant la valeur manquante, et reprend plus loin.

On peut également jouer des instructions `PLOT` et `PLOT2` pour afficher, comme sur la figure 4.17, la courbe interrompue au premier plan (avec `PLOT2`) et en dessous une courbe pointillée continue (avec `PLOT`) qu'on ne verra que dans les intervalles dus aux interruptions.

Exemple 4.13 – L'option SKIPMISS pour les séries incomplètes

```
DATA work.essence ;
       SET sashelp.citimon
            (WHERE = (date BETWEEN "01jan1980"d AND "31dec1981"d)) ;
       IF MONTH(date) IN (7,8) THEN eegp = . ;
       LABEL eegp = "Prix de l'essence" ;
RUN ;
SYMBOL1 I = JOIN C = GREEN L = 3 ;
SYMBOL2 I = JOIN C = BLUE W = 2 ;
PROC GPLOT DATA = work.essence ;
       PLOT eegp * date ;
       PLOT2 eegp * date / SKIPMISS ;
RUN ; QUIT ;
```

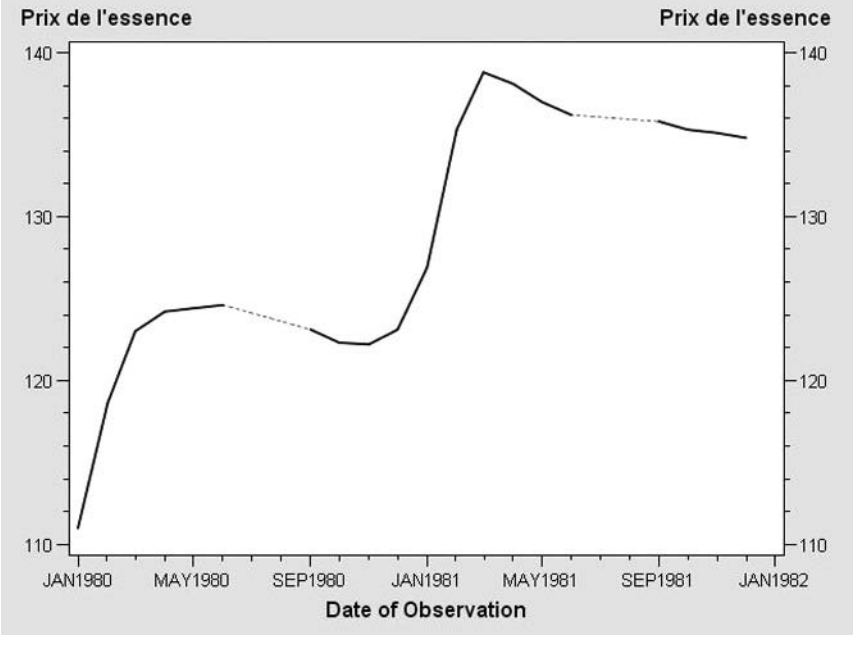


Figure 4.17 — Résultat de l'exemple 4.13 (driver ActiveX)

4.5.5 Les graphiques à bulles

Les graphiques à bulles sont un moyen de représenter simultanément trois variables dont au moins deux sont quantitatives, sous une forme lisible : les points y sont représentés sous forme de cercles de rayon proportionnel aux valeurs d'une des variables quantitatives. Il n'est possible dans ce cas de superposer plusieurs séries de points qu'en utilisant un système de double échelle semblable au principe de `PLOT` / `PLOT2` décrit à la section précédente.

Si la variable contrôlant la taille des bulles a des valeurs positives, le cercle est tracé en ligne pleine. Si les valeurs sont négatives, c'est un cercle pointillé qui s'affiche.

```
SYMBOL I = typeCourbe V = marqueur POINTLABEL = ("#variable") ;
PROC PLOT DATA = tableSAS ;
   BUBBLE  vary1 * varX = varTaille1 < / HREF=valeur VREF=valeur
                                        BLABEL BSIZE=multiplicateur > ;
 < BUBBLE2 vary2 * varX = varTaille2 < / HREF=valeur VREF=valeur
                                        BLABEL BSIZE=multiplicateur > ;
RUN ; QUIT ;
```

L'option `BLABEL` indique dans ou à côté de chaque bulle la valeur associée à sa taille, tandis que l'option `BSIZE` permet de faire intervenir un facteur multiplicatif dans le calcul de la taille (indispensable quand on rend compte, par la taille des bulles, de très fortes valeurs).

Exemple 4.14 – Graphique à bulles

```
ODS OUTPUT summary = work.stats ;
PROC MEANS DATA = livre.voitures MEAN ;
       VAR cylindree ;
       CLASS type ;
RUN ;
PROC GPLOT DATA = work.stats ;
       BUBBLE cylindree_mean * type = nobs / BLABEL ;
RUN ; QUIT ;
```

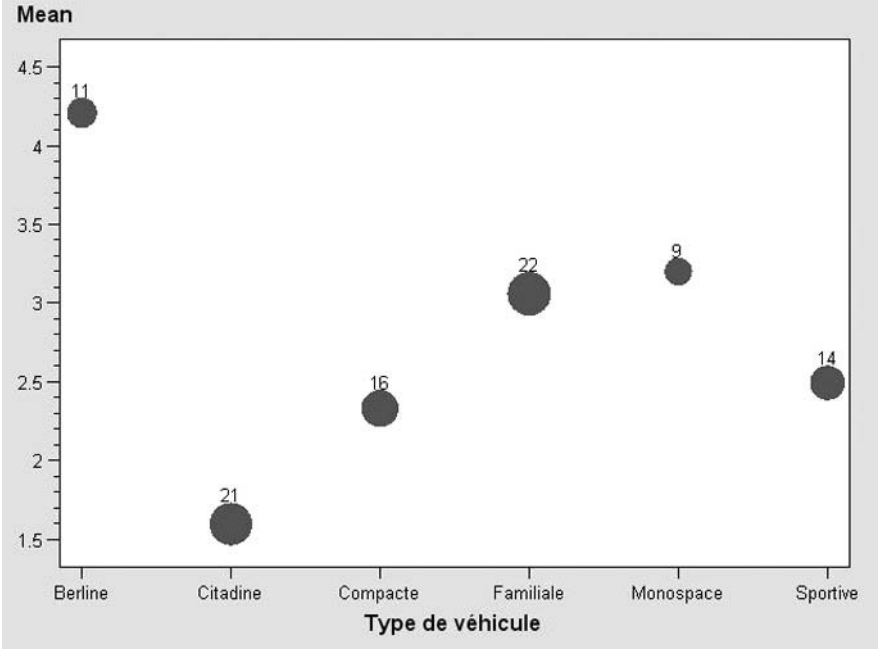


Figure 4.18 — Résultat de l'exemple 4.14 (driver ActiveX)

4.5.6 Les courbes de densité, alternative aux histogrammes

Comment superposer la distribution d'une variable continue pour plusieurs groupes d'observations ? Les histogrammes ne se superposent pas aisément, et il est malaisé de comparer deux graphiques, fussent-ils côte à côte. Il est possible de superposer des courbes de densité, semblables à celles construites par la procédure Univariate : à chaque valeur de la variable représentée, on calcule la fréquence et on présente le tout sous une forme lissée. Ce calcul préalable (lissage compris) peut être réalisé par la procédure KDE ; le rendu graphique est une courbe simple, *via* la procédure GPLOT.

La syntaxe de la procédure KDE change entre les versions 8 ou 9 de SAS (dans l'exemple 4.15, la syntaxe de la version 8 apparaît en commentaire). Comme la procédure KDE contient une instruction `BY`, il est nécessaire que la table lue ait été triée ou indexée en amont.

Exemple 4.15 – Courbes de densité superposées

```
PROC SORT DATA = livre.voitures
           OUT = work.voitures ;
  BY type ;
RUN ;
PROC KDE DATA = work.voitures ;
  UNIVAR cylindree / OUT = work.densites ;
  BY type ;
RUN ;
/* syntaxe de SAS version 8…
PROC KDE DATA = work.voitures
          OUT = work.densites ;
  VAR cylindree ;
  BY type ;
RUN ;
*/
SYMBOL i = join v = none ;
PROC GPLOT DATA = work.densites ;
  PLOT density * value = type ;
RUN ; QUIT ;
```

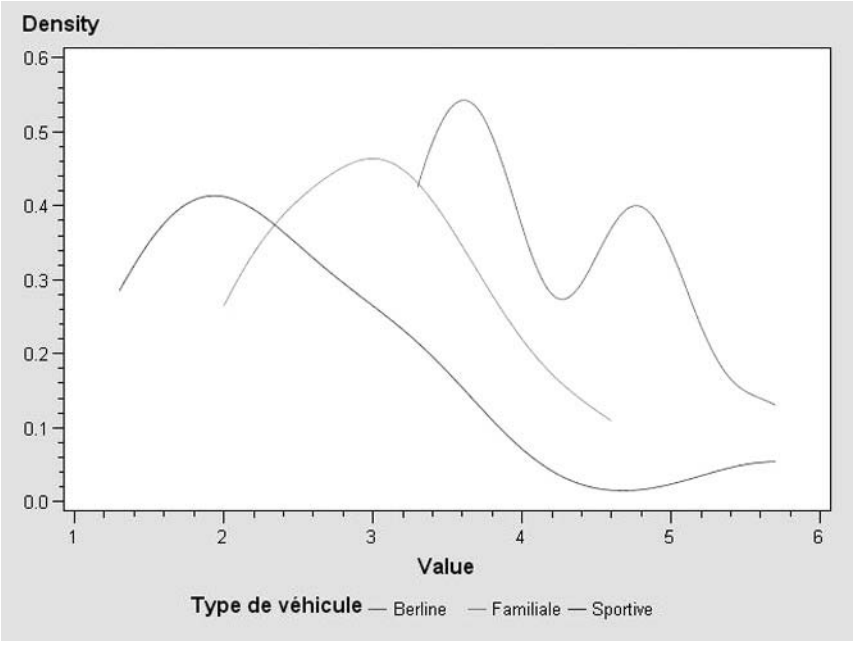


Figure 4.19 — Résultat de l'exemple 4.15 (driver ActiveX)

4.6 LES COULEURS, LES LÉGENDES, LES AXES

Comme les instructions `PATTERN` et `SYMBOL` déjà détaillées dans les procédures GCHART et GPLOT respectivement, d'autres instructions permettent de personnaliser les légendes et les axes d'un graphique. Leur syntaxe est identique quelle que soit la procédure à laquelle elles sont destinées ; on verra pour chacune comment y faire référence explicitement dans une procédure GCHART ou GPLOT.

4.6.1 La construction d'une légende

Il est possible de définir jusqu'à 99 légendes différentes simultanément, en jouant sur le numéro collé à l'instruction : `LEGEND1` à `LEGEND99`.

```
LEGENDn < FRAME >
     POSITION=(LEFT|CENTER|RIGHT TOP|MIDDLE|BOTTOM INSIDE|OUTSIDE)
     < ACROSS=nbColonnes DOWN=nbLignes >
     < LABEL="texte en tête de légende"|NONE >
     < ORDER=(valeur1 < valeur2 < … > >) > ;
PROC GPLOT … ;
   PLOT|BUBBLE|PLOT2|BUBBLE2 … / LEGEND = legendn ;
RUN ; QUIT ;
PROC GCHART … ;
```

```
   HBAR|VBAR|PIE|HBAR3D|VBAR3D|PIE3D … / LEGEND = legendn ;
RUN ; QUIT ;
```

L'option FRAME permet d'avoir une bordure autour de la légende (par défaut, il n'y en a pas). POSITION indique l'emplacement de la légende (INSIDE ou OUTSIDE la situent à l'intérieur ou à l'extérieur du graphique). ACROSS et DOWN indiquent le nombre de valeurs par ligne et le nombre de lignes de la légende, respectivement. LABEL est le texte figurant en titre de la légende (NONE : aucun titre). Enfin ORDER énumère quelles valeurs sont des entrées de la légende et dans quel ordre.

On fait appel à la légende dans la procédure GCHART comme dans GPLOT : en option, dans les instructions de dessin (HBAR, VBAR, PIE, PLOT, BUBBLE).

Exemple 4.16 – Ajout d'une légende à un graphique à deux échelles

```
PROC SQL ;
  CREATE TABLE work.stats AS
    SELECT puissance,
           SUM(ventes) AS Tventes,
           MEAN(poids*1000) AS Mpoids
    FROM livre.voitures
    GROUP BY puissance ;
QUIT ;
DATA work.stats ;
  SET work.stats ;
  serie = "Ventes totales  " ;
  ventes = Tventes ;
  OUTPUT ;
  serie = "Poids moyen (kg)" ;
  poids = Mpoids ;
  OUTPUT ;
RUN ;
SYMBOL1 i = join w = 2 ;
SYMBOL2 i = join w = 2 ;
LEGEND1 LABEL=NONE DOWN=2 POSITION=(INSIDE TOP LEFT) FRAME ;
PROC GPLOT DATA = work.stats ;
  PLOT ventes * puissance = serie / LEGEND=legend1 ;
  PLOT2 poids * puissance = serie / LEGEND=legend1 ;
RUN ; QUIT ;
```

L'exemple 4.16 est une reprise de l'exemple 4.12 sur les graphiques à double échelle. En ajoutant une légende, on peut enfin savoir quelle courbe correspond à quelle information. Cependant, un travail sur les données est nécessaire au préalable pour pouvoir prétendre à l'affichage d'une légende (SAS s'y refuse s'il ne trace qu'une courbe par échelle, d'où la nécessité de pouvoir tracer avec la syntaxe varY*varX=groupe).

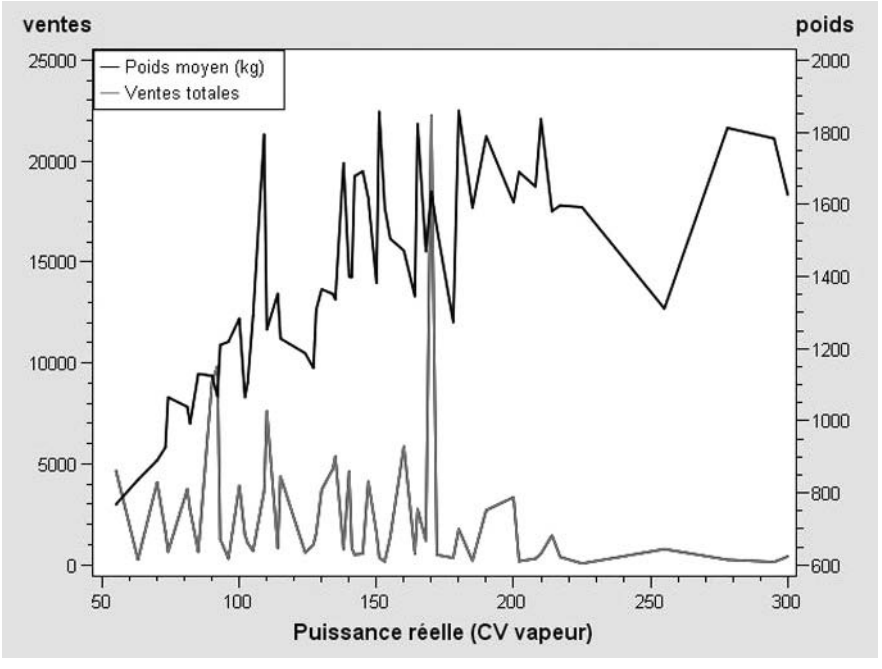


Figure 4.20 — Résultat de l'exemple 4.16 (driver ActiveX)

4.6.2 La manipulation des axes

On peut définir jusqu'à 99 axes différents avec les instructions `AXIS1` jusqu'à `AXIS99`. On fait appel à l'une ou l'autre des définitions avec les options `HAXIS` et `VAXIS` (axe des abscisses et des ordonnées) dans la procédure GPLOT, et avec les options `MAXIS`, `RAXIS` et `GAXIS` (axe de la variable représentée, de la statistique et d'éventuels groupes) dans la procédure GCHART.

```
AXISn LABEL=(option(s) "texte1" < option(s) "texte2" < … > >)
      ORDER=(valeur(s)) VALUE=(option(s) valeur(s))
      LOGBASE=10 LOGSTYLE=EXPAND|POWER ;
PROC GPLOT … ;
   PLOT|PLOT2|BUBBLE|BUBBLE2 … / VAXIS=axisn | HAXIS=axisn ;
RUN ; QUIT ;
PROC GCHART … ;
   HBAR|VBAR|HBAR3D|VBAR3D … / RAXIS=axisn | MAXIS=axisn | GAXIS=axisn ;
RUN ; QUIT ;
```

Il est possible d'utiliser des échelles logarithmiques : pour cela, on doit utiliser les options `LOGBASE=10` et `LOGSTYLE=EXPAND` (qui affiche les valeurs telles quelles, de préférence à `LOGSTYLE=POWER` qui les affiche en puissances de 10).

L'option `LABEL` permet d'ajouter un titre à l'axe ; diverses options de mise en forme peuvent précéder tout ou partie de ce texte.

On retrouve ces mêmes indications de mise en forme dans l'option VALUE, concernant cette fois les graduations de l'axe. Parmi ces sous-options de mise en forme, notons tout d'abord JUSTIFY=LEFT|CENTER|RIGHT pour l'alignement de ce texte. Les deux sous-options ANGLE et ROTATE sont complémentaires : ANGLE indique à quel angle (en degrés ; le sens positif est celui inverse des aiguilles d'une montre) se situe la ligne de base de texte ; ROTATE indique à quel angle sont écrits les caractères par rapport à la ligne de base.

En plus de l'option VALUE qui indique comment afficher les valeurs, l'option ORDER permet de définir les graduations de l'axe. Elle est particulièrement utile pour les axes rendant des valeurs continues : on définit alors sous forme d'intervalle le minimum, le maximum et l'intervalle des graduations de cet axe. L'écriture est ORDER=(min TO max BY intervalle).

Exemple 4.17 – Personnalisation des axes d'un histogramme

```
AXIS1 LABEL=(ROTATE=90 ANGLE=-90 "Nombre de modèles")
      ORDER=(0 TO 30 BY 10)
      VALUE=("0%" "10%" "20%" "30%") ;
AXIS2 LABEL=NONE VALUE=(ANGLE=20) ;
PROC GCHART DATA = livre.voitures ;
  VBAR type / SUBGROUP=transmission
                TYPE=PERCENT
               MAXIS=axis2
               RAXIS=axis1 ;
RUN ; QUIT ;
```

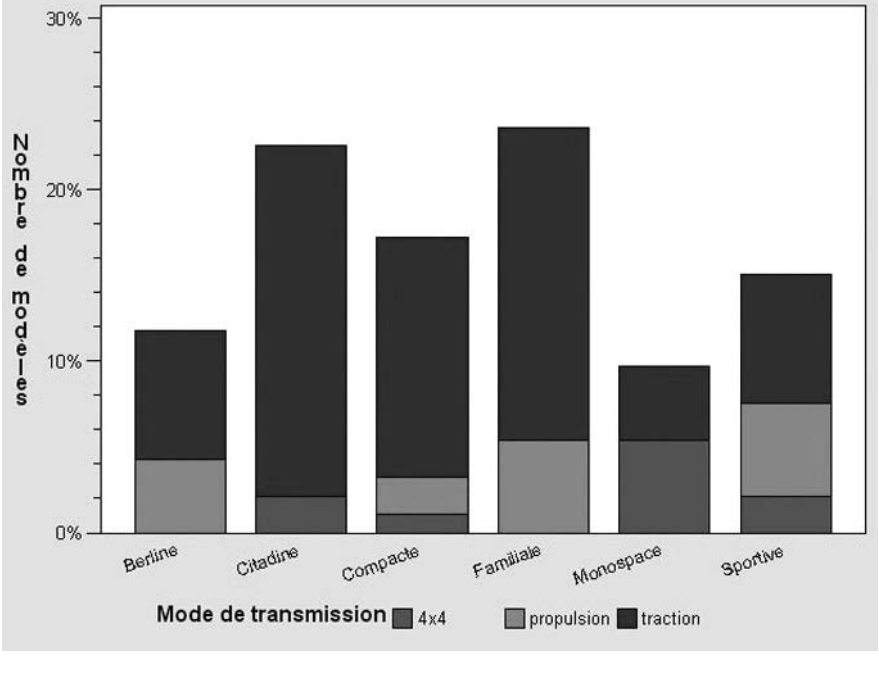


Figure 4.21 — Résultat de l'exemple 4.17 (driver ActiveX)

4.7 LES AJOUTS À FAÇON SUPERPOSÉS À UN GRAPHIQUE (TABLES *ANNOTATE*)

Pour améliorer un graphique SAS, il n'est forcément payant de passer des heures à chercher quelque option magique. Il est souvent plus simple de donner à SAS des instructions de dessin pour qu'il surcharge la version « brute » du graphique. Ces instructions de dessin (ajouter du texte, des traits, etc.) sont consignées dans une table SAS dont le nom est libre (même si on parle souvent de *table Annotate*) ; on la construit à l'aide d'une étape Data. Elle est ensuite appelée dans la procédure graphique concernée : GPLOT, GCHART, mais aussi BOXPLOT et UNIVARIATE, dans une option `ANNOTATE` de l'instruction graphique (`PLOT`, `BUBBLE`, `VBAR`, `HBAR`, `PIE`, `HISTOGRAM`, etc.).

4.7.1 Le concept d'Annotate

La table Annotate décrit l'ensemble des actions à mener pour ajouter des éléments à un graphique produit par une procédure. Il s'agit d'une table SAS tout à fait normale, où chaque observation correspond à une action à effectuer. Les variables de cette table ont des noms normalisés : parmi elles, c'est la variable FUNCTION qui indique le type d'élément graphique à ajouter et influe sur l'interprétation des autres (tableau 4.3).

Les variables X et Y indiquent des positions horizontale et verticale dans le graphique. X et Y ne peuvent avoir que des valeurs numériques ; si le graphique a un axe horizontal associé à une variable de type caractère, alors il faudra utiliser une variable XC de type texte.

Les variables XSYS et YSYS indiquent dans quelle unité sont données les valeurs des variables X/XC et Y. On utilise le plus souvent la valeur `"2"` (XSYS et YSYS sont obligatoirement de type caractère) qui fait référence aux unités des axes du graphique. On peut aussi utiliser la valeur `"3"` qui exprime X et Y en pourcentages de l'ensemble de la fenêtre graphique.

WHEN indique quand l'élément doit être ajouté au graphique : avant (valeur `"B"`) ou après (valeur `"A"`, la plus courante) le dessin par la procédure graphique.

Tableau 4.3 — Principales valeurs de la variable `FUNCTION`, variables associées

`FUNCTION` vaut…	Action	Variables associées	Usage
`BAR`	Dessiner un rectangle	X, XC, Y	Coin supérieur droit
		COLOR	Couleur de la barre, de la bordure
		STYLE	Vaut `E` si la barre est vide, `S` sinon
`DRAW`	Tracer un segment	X, XC, Y	Fin du segment

FUNCTION vaut…	Action	Variables associées	Usage
LABEL ou SYMBOL	Afficher un texte	X, XC, Y	Emplacement du texte
		TEXT	Texte à afficher
		POSITION	Alignement du texte (cf. infra)
		SIZE	Taille du texte
MOVE	Déplacer le curseur graphique sans tracer quoi que ce soit	X, XC, Y	Point d'arrivée du curseur
PIE	Dessiner un arc de cercle	X, XC, Y	Centre de l'arc de cercle
		ROTATE	Angle (en degrés) de l'arc
		ANGLE	Angle (en degrés) du point d'où part l'arc de cercles par rapport à l'horizontale
		SIZE	Rayon du cercle (pour son unité, dépend de la variable HSYS qui vaut souvent "2" : unité des axes)
		LINE	Choisit si on dessine un arc de cercle sans rayons ou avec

4.7.2 L'ajout de texte à un graphique

La fonction associée à l'ajout de texte est LABEL. On indique le texte à afficher dans la variable TEXT, tandis que SIZE en régit la taille. L'emplacement du texte est plus subtil : s'il est situé par défaut au point de coordonnées X (ou XC) et Y, il peut être décalé selon la valeur de la variable POSITION, de 1 à 9 et de A à F.

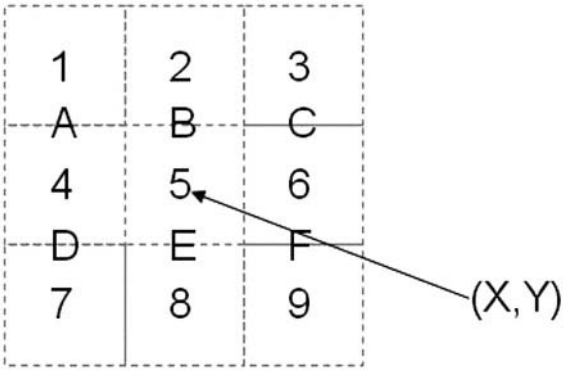


Figure 4.22 — Valeurs possibles de la variable POSITION

Dans l'exemple 4.18, un diagramme en bâtons indiquant la consommation moyenne des véhicules est enrichi du nombre de modèles par type.

Exemple 4.18 – Table Annotate pour ajouter du texte à un graphique

```
ODS OUTPUT summary = work.stats ;
PROC MEANS DATA = livre.voitures MEAN ;
       VAR conso_ville ;
       CLASS type ;
RUN ;
DATA work.stats ;
       SET work.stats ;
       xsys = "2" ; ysys = "2" ;
       when = "A" ; function = "LABEL" ;
       xc = type ; y = conso_ville_mean ; /* coordonnées */
       text = COMPRESS(nobs)!!" modèles" ; color = "WHITE" ;
       position = "8" ; /* légèrement sous le point (Xc,Y), centré */
RUN ;
AXIS1 LABEL=(ANGLE=90) ;
PROC GCHART DATA = livre.voitures ;
       VBAR type / SUMVAR=conso_ville TYPE=MEAN RAXIS=axis1
                  ANNOTATE=work.stats ;
RUN ; QUIT ;
```

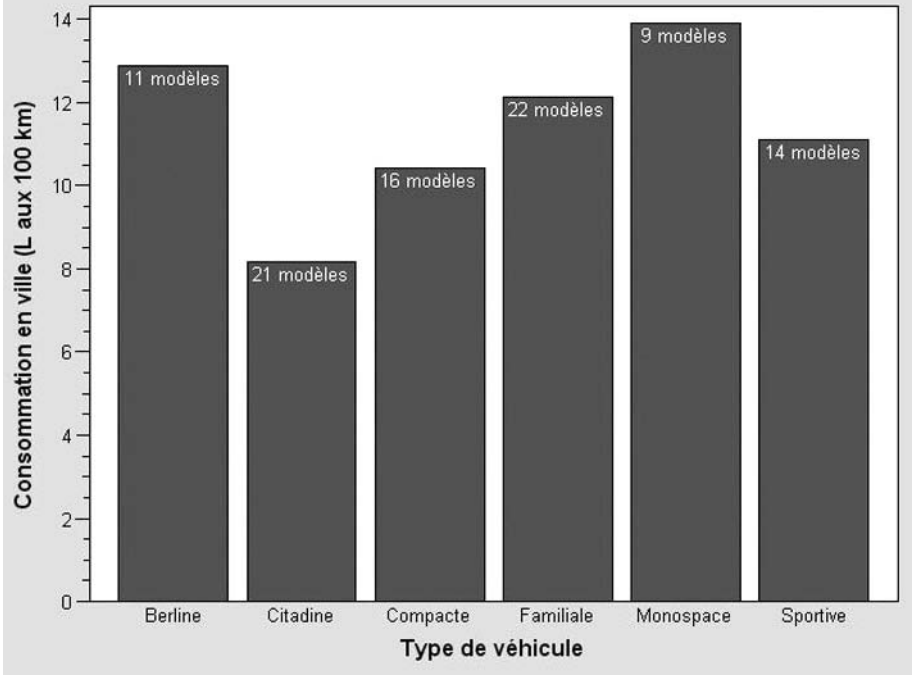


Figure 4.23 — Résultat de l'exemple 4.18 (driver ActiveX)

4.7.3 L'ajout de traits sur un graphique

Deux fonctions sont souvent nécessaires pour ajouter un segment à un graphique : DRAW, bien sûr, mais alors les coordonnées dans les variables X et Y (ou XC et Y) n'indiquent qu'une des extrémités du trait. L'autre correspond à l'emplacement précédent du curseur. Donc pour tracer des segments déconnectés les uns des autres, on alterne les fonctions MOVE et DRAW. Dans le cas où les segments s'enchaînent, en revanche, il n'est pas nécessaire d'introduire de fonction MOVE, sauf pour le premier des traits.

L'exemple 4.19 est lié à une analyse statistique connue sous le nom d'analyse en composantes principales (ou ACP). Elle est réalisée par la procédure Princomp, qui permet de récupérer les coordonnées de vecteurs. Ceux-ci sont d'ordinaire représentés par des flèches allant du point (0,0) à leur autre extrémité. À chaque flèche correspond une variable du fichier d'origine. Toutes ces flèches sont situées à l'intérieur d'un cercle de rayon 1 et de centre (0,0) qu'il est agréable de réussir à afficher.

Exemple 4.19 – Table Annotate pour l'ACP

```
ODS OUTPUT eigenVectors = work.planFactoriel ;
PROC PRINCOMP DATA = livre.voitures ;
       VAR longueur poids cylindree puissance vol_coffre ;
RUN ;
DATA work.decoration ;
       SET work.planFactoriel ;
       xsys = "2" ; ysys = "2" ; when = "A" ;
       IF _N_=1 THEN DO ; /* tracé du cercle */
              x = 1 ; y = 0 ; function = "MOVE " ;
              OUTPUT ;
              function = "DRAW " ;
              DO angle = 0 TO 2*3.14 BY 0.04 ;
                     x = COS(angle) ; y = SIN(angle) ;
                     OUTPUT ;
              END ;
       END ;
       x = 0 ; y = 0 ; function = "MOVE " ;
       OUTPUT ;
       x = prin1 ; y = prin2 ; function = "DRAW " ;
       OUTPUT ; /* tracé des vecteurs */
       function = "LABEL" ; text = variable ; position = "6" ;
       OUTPUT ; /* ajout des labels */
RUN ;
SYMBOL i = none v = triangle ;
AXIS1 ORDER=(-1 TO 1 BY 0.5) ;
PROC GPLOT DATA = work.planFactoriel ;
       PLOT prin2 * prin1 / HREF=0 HAXIS=axis1
                            VREF=0 VAXIS=axis1
                          ANNOTATE=work.decoration ;
RUN ; QUIT ;
```

La table Annotate contient trois séries d'éléments : d'abord le cercle (qui s'avère être un ovale vu que les deux axes ne sont pas d'égale longueur, ce qui empêche de

générer ce cercle avec une fonction PIE) défini par une série de segments, puis des couples MOVE et DRAW pour tracer les vecteurs, et enfin les labels affichés à la droite des vecteurs.

Elle enrichit considérablement la sortie de GPLOT qui se contente de placer les triangles sur le graphique.

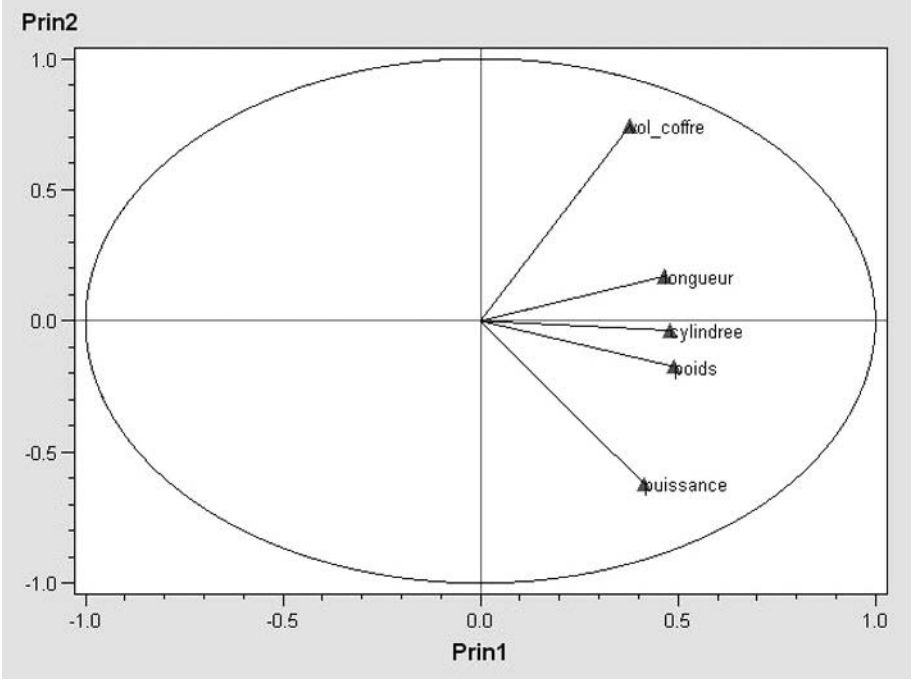


Figure 4.24 — Résultat de l'exemple 4.19 (driver ActiveX)

4.8 LA FUSION DE PLUSIEURS GRAPHIQUES EN UNE SEULE IMAGE

Il est possible, avec la procédure GREPLAY de SAS/GRAPH, de fusionner plusieurs graphiques en une seule image. Pour cela, il faut avoir sauvegardé les différents graphiques dans un catalogue et les intégrer dans un canevas (appelé *template*) prédéfini. Ces *templates* graphiques n'ont aucun rapport, fors leur nom, avec les *templates* (ou modèles) de l'ODS détaillés au chapitre 5.

4.8.1 La sauvegarde d'un graphique

Par défaut, les graphiques sont conservés dans le catalogue WORK.GSEG jusqu'à la fin de la session SAS. Ils portent des noms automatiques qui traduisent la procédure

les ayant générés : GPLOT, puis GPLOT1, GPLOT2, etc. par exemple. Il est également possible de sauvegarder explicitement un graphique : on indiquera dans l'instruction démarrant la procédure graphique le catalogue de stockage *via* l'option `GOUT`, puis le nom du graphique dans l'instruction de dessin (`HBAR`, `VBAR`, `PLOT`, etc.) via l'option `NAME`.

Ce nom ne doit pas dépasser 8 caractères (il sera tronqué sinon) et ne doit être composé que des caractères habituellement utilisés pour les noms SAS (caractères non accentués, chiffres mais pas en premier, underscore).

```
PROC GPLOT|GCHART … GOUT = bibliotheque.catalogueSauvegarde ;
  PLOT|BUBBLE|HBAR|VBAR|PIE … / NAME = "nomGraphique" ;
RUN ; QUIT ;
```

4.8.2 L'utilisation d'un canevas prédéfini

S'il est possible de définir des *templates* sur mesure pour y intégrer ses graphiques, nous nous limiterons ici à décrire l'utilisation de ceux qui sont prédéfinis dans SAS. On les trouve dans le catalogue SASHELP.TEMPLT.

```
PROC GREPLAY NOFS IGOUT = bibliotheque.catalogueSauvegarde
                     TC = sashelp.templt|autreCatalogueTemplates
                TEMPLATE = nomTemplate ;
   TREPLAY 1: nomGraphique1
      <    2: nomGraphique2  < … > > ;
RUN ; QUIT ;
```

Leurs noms sont des indications sur leur organisation : H2 désigne un *template* avec deux graphiques côte à côte (H pour *Horizontal*) ; V2 présente deux graphiques l'un au-dessus de l'autre (V pour *Vertical*). Leurs variantes H2S et V2S proposent une marge (S pour *Space*) entre les deux graphiques. Sur cette logique, on trouve encore les *templates* H3 et H4, H3S et H4S, V3 et V3S.

Autre logique dans les noms, les compositions comme L1R2 proposent un graphique à gauche (L pour *Left*) et deux à droite (R pour *Right*). Également disponibles : L1R2S, L2R1 et L2R1S, L2R2 (quatre graphiques en deux colonnes de deux) et L2R2S. Enfin, on trouve les *templates* comme U1D2 avec un graphique en haut (U pour *Up*) et deux en bas (D pour *Down*) ; U1D2S, U2D1 et U2D1S en sont des variantes.

Dans l'exemple 4.20, on utilise les graphiques produits dans l'exemple 4.11, et on les organise dans le *template* L2R2S.

Exemple 4.20 – Fusion de trois graphiques en une seule image

```
PROC GREPLAY NOFS IGOUT = work.gseg
                     TC = sashelp.templt
                TEMPLATE = L1R2S ;
  TREPLAY 1: gplot1 2: gplot 3: gplot2 ;
RUN ; QUIT ;
```

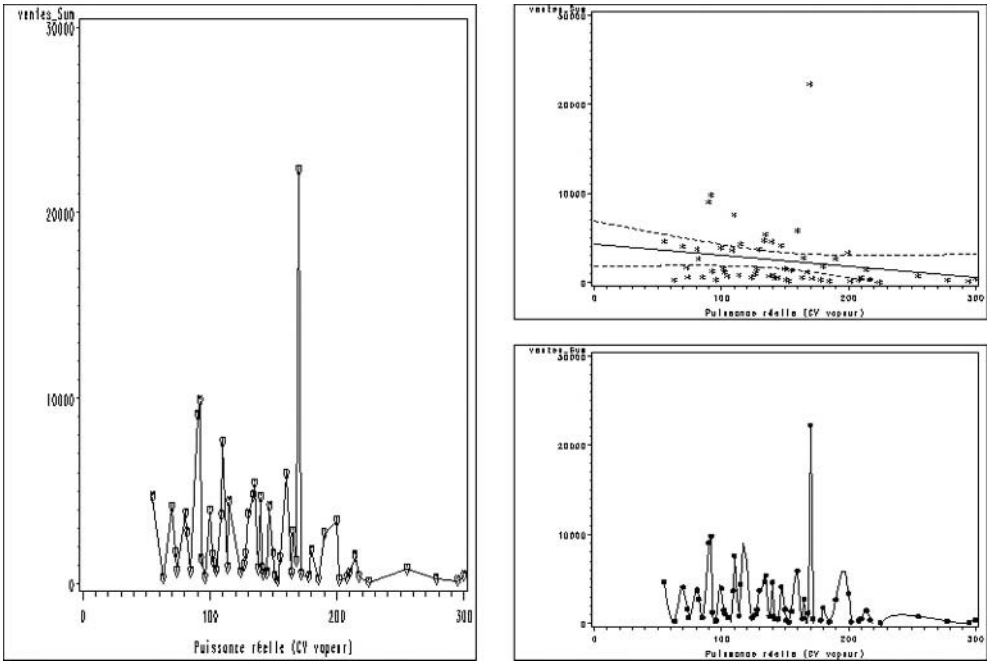


Figure 4.25 — Résultat de l'exemple 4.20 (driver ActiveX)

4.8.3 Les limites de ce type de fusion d'images

Malheureusement, la procédure GREPLAY n'est pas compatible (au moins jusqu'à SAS version 9.1.3) avec les drivers ActiveX, Java et leurs dérivés. Les graphiques fusionnés par cette procédure gardent donc un aspect semblable à ce qu'ils ont dans la fenêtre GRAPH de SAS.

Deux solutions (expérimentales dans la version 9.1.3) permettent de présenter côte à côte plusieurs graphiques sans renoncer à une esthétique très soignée : le GTL qui est capable de produire, *via* un driver Java, des graphiques fusionnés (mais pas de diagrammes circulaires ou de graphiques en bulles) ; et l'insertion d'images dans un tableau construit par l'objet ODSOUT, ce qui ne mène pas réellement à une fusion d'images, mais juste à une présentation juxtaposée. Ces fonctionnalités seront décrites au chapitre 8.

4.9 LES GRAPHIQUES STATISTIQUES *VIA* L'ODS

Expérimental en version 9.1, `ODS GRAPHICS` est un système permettant l'édition de graphiques à l'utilisation d'une procédure statistique. Les procédures de SAS/GRAPH telles que GPLOT et GCHART ne sont pas mises en œuvre pour créer les graphiques, qui sont générés par un driver Java indépendant. Cependant, une licence pour le module SAS/GRAPH sera nécessaire à ces fonctionnalités.

Les procédures qui éditent spontanément de tels graphiques sont de plus en plus nombreuses à chaque version de SAS. Il y en a une trentaine en version 9.1.3. Quelques exemples suivent ; pour les autres, se référer à la documentation de chaque procédure – leur spécialisation statistique les met pour la plupart largement au-delà du sujet de ce livre.

Afin de décrire comment les graphiques sont générés, SAS propose un langage spécifique, le GTL. Il est accessible à travers la procédure TEMPLATE et permet aussi bien de modifier les graphiques générés par les procédures que de se créer ses propres graphiques : nous évoquerons dans le chapitre 8 cette dernière fonctionnalité, car c'est une alternative très intéressante aux procédures précédentes, et spécialement GREPLAY. La version 9.2 de SAS verra de plus la création de nouvelles procédures utilisant le GTL, dont la procédure SGRENDER chargée de produire tel graphique décrit en GTL.

Dans SAS version 9.1, les procédures suivantes peuvent produire des graphiques à travers le système ODS GRAPHICS : CORR[1] (module de base), ARIMA, AUTOREG, ENTROPY, EXPAND, MODEL, SYSLIN, TIMESERIES, UCM, VARMAX, X12 (module SAS/ETS), HPF (module HPF ou Forecast Server), ANOVA, CORRESP, GAM, GENMOD, GLM, KDE, LIFETEST, LOESS, LOGISTIC, MI, MIXED, PHREG, PRINCOMP, PRINQUAL, REG et ROBUSTREG (module SAS/STAT).

Pour produire ces graphiques, il faut envoyer les résultats dans une destination ODS compatible (HTML, RTF, PDF) et activer ODS GRAPHICS.

```
ODS GRAPHICS ON < / IMAGEFMT = GIF | JPEG | PNG | STATICMAP
                   IMAGENAME = "nomFichierImage" RESET > ;
```

Les options n'ont d'effet qu'en conjonction avec la destination ODS HTML. `IMAGEFMT` permet de choisir le type d'image, ou de demander des info-bulles automatiques (`IMAGEFMT=STATICMAP`). `IMAGENAME` force le nom des fichiers images produits : à ce nom seront suffixés un numéro puis l'extension .GIF, .JPG ou .PNG. L'option `RESET` permet de remettre le numéro suffixé à 0.

Une instruction `ODS GRAPHICS OFF ;` sans option permet de désactiver la production de graphiques.

Exemple 4.21 – Production de graphiques à partir d'une procédure GLM

```
ODS GRAPHICS ON / IMAGEFMT=STATICMAP ;
ODS HTML FILE="c:\temp\anova.htm" GPATH="c:\temp" ;
PROC GLM DATA = livre.voitures ;
       CLASS type ;
       MODEL poids = type ;
RUN ; QUIT ;
ODS HTML CLOSE ;
```

1. Un exemple de graphique produit par la procédure CORR est donné dans *SAS – Maîtriser SAS Base et SAS Macro*, op. cit., page 205.

Dans les sorties de la procédure, les tableaux habituels sont suivis d'un graphique, qui bénéficie grâce à `IMAGEFMT=STATICMAP` d'info-bulles quand on arrête la souris sur un des éléments.

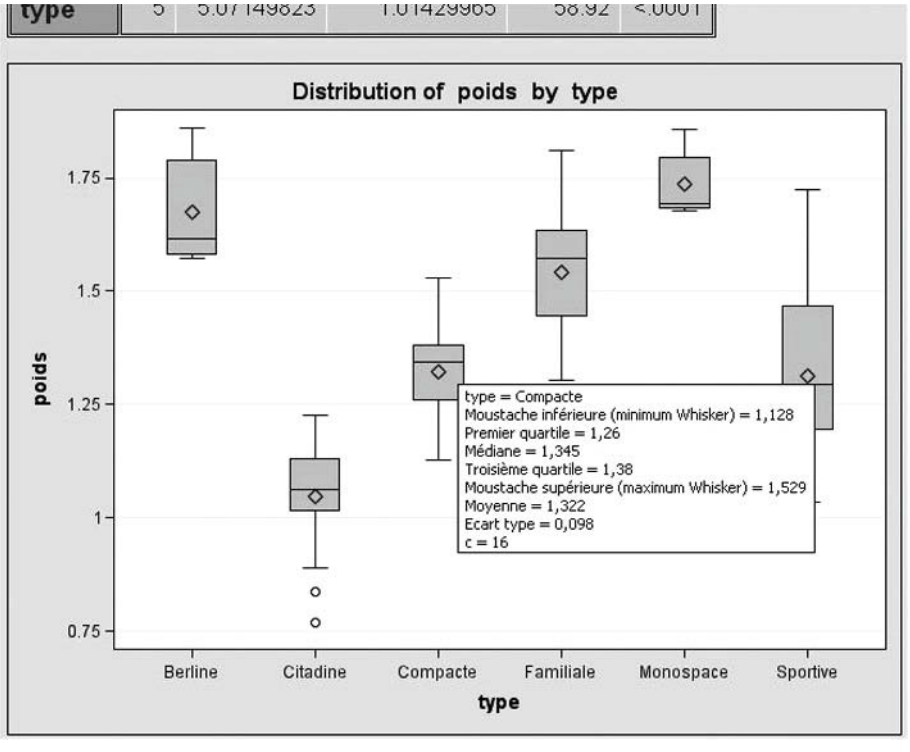


Figure 4.26 — Résultat de l'exemple 4.21 (driver Java implicite d'ODS GRAPHICS)

5

La gestion des mises en forme

Objectifs

Ce chapitre décrit l'utilisation de la procédure TEMPLATE et la notion de modèles. Ceux-ci agissent à deux niveaux : un modèle tabulaire et un modèle de style. En créant ses propres modèles, on peut aussi bien modifier l'aspect global de toutes les sorties des procédures (en créant un modèle de style) que toucher à l'aspect d'un objet ODS spécifique (en modifiant son modèle tabulaire).

5.1 LA NOTION DE MODÈLE OU *TEMPLATE*

Le chapitre 1 explique comment les objets ODS sont au cœur de la transcription de sorties SAS en documents extérieurs ; ils portent l'information produite par la procédure jusqu'à l'ODS qui se charge de sa mise en forme. Pour assurer cette tâche, deux modèles (ou *templates*) sont mobilisés : l'un est *spécifique à l'objet* (le modèle tabulaire) et l'autre est *global* à la destination ODS (le modèle de style).

5.1.1 Les modèles de style et les modèles tabulaires

Certaines procédures (PRINT, TABULATE, REPORT, mais aussi FREQ pour les tableaux croisés) n'ont pas de modèle tabulaire. Les manipulations de la section 5.2 ne concernent donc pas ces sorties. Pour les autres procédures, on connaîtra le nom

et l'emplacement du modèle utilisé par l'intermédiaire de l'instruction `ODS TRACE` décrite à la section 1.1.1., dans la ligne *`TEMPLATE`*.

Exemple 5.01 – Quel modèle tabulaire est associé à une procédure ?

```
ODS TRACE ON ;
PROC CONTENTS DATA = livre.voitures ;
RUN ;
ODS TRACE OFF ;
```

Dans la fenêtre Log, on lit entre autres…

```
Output Added:
-------------
Name:       Attributes
Label:      Attributes
Template:   Base.Contents.Attributes
Path:       Contents.DataSet.Attributes
-------------
```

À l'objet ATTRIBUTES est associé un modèle tabulaire stocké à l'emplacement `BASE.CONTENTS.ATTRIBUTES`. La suite indique quel est l'emplacement physique de l'objet.

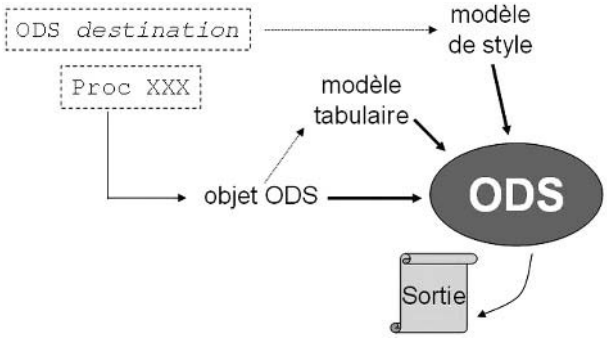


Figure 5.1 — Les différents modèles dans la construction d'une sortie ODS

5.1.2 Le stockage des modèles

Les modèles sont stockés dans des fichiers appelés *items stores* (ce qu'on pourrait traduire par « magasins d'items » ou « entrepôts d'articles »). Ils sont un équivalent des catalogues : là où le système voit un fichier unique, SAS peut l'organiser de manière virtuelle (= logique, et non physique) en sous-répertoires et y ranger plusieurs éléments. Le stockage par défaut des modèles est SASUSER.TEMPLAT ; il est lié à l'emplacement de la bibliothèque SASUSER. SAS propose également des modèles prédéfinis dans SASHELP.TMPLMST. Il est possible d'indiquer tout autre emplacement quand on définit un modèle avec la procédure TEMPLATE, *via* l'option `STORE` dans l'instruction `DEFINE`.

Pour s'assurer que l'on va chercher le modèle au bon emplacement, on peut utiliser l'instruction ODS PATH pour indiquer dans quels items stores (et dans quel ordre) on recherche des modèles.

```
ODS PATH sasuser.templat (WRITE)
       < bibliothèque.itemsStore (READ|WRITE) >
         sashelp.tmplmst (READ) ;
```

L'ordre des chemins proposés dans ODS PATH est important : en cas d'homonymie, SAS utilisera le premier modèle trouvé en parcourant les items stores dans cet ordre. Il n'est donc pas forcément judicieux de commencer la liste par celui de SASHELP si on veut en surcharger une définition. La valeur d'ODS PATH est retenue pour la durée de la session SAS. Par défaut, elle correspond à la séquence SASUSER.TEMPLAT puis SASHELP.TMPLMST.

On peut indiquer à côté de chaque items store s'il est en lecture seule ou non, à l'aide des mots-clés READ pour la lecture seule et WRITE pour la lecture/écriture.

Les modèles tabulaires sont stockés dans des dossiers aux noms variables ; certains ont trait au nom du module de la procédure, mais ce n'est pas une règle absolue.

Les modèles de style sont stockés dans le dossier STYLES. Cette règle ne souffre pas d'exception – sans quoi SAS sera incapable d'utiliser le modèle.

On trouve encore, dans le dossier TAGSETS, des jeux de balises pour permettre une transcription des sorties SAS dans des formats de fichiers qui s'y prêtent : HTML, CSV, XML, mais également RTF, TROFF, PMML, etc. La définition de tels jeux de balises ne sera pas abordée dans cet ouvrage.

5.2 LA PERSONNALISATION D'UN MODÈLE TABULAIRE

La possibilité de créer de toutes pièces un modèle tabulaire sera explorée à la section 5.2.4 en tant qu'alternative aux procédures Print et Report.

Mais le plus souvent, on ne cherche pas à créer *ex nihilo* un modèle tabulaire ; on s'appuie sur un modèle existant pour le modifier. On intègre alors dans la procédure TEMPLATE une syntaxe commençant par EDIT et terminée par END. À l'intérieur de cette redéfinition du modèle, on pourra définir ou redéfinir la liste des variables (COLUMN), des en-têtes (DEFINE HEADER), des pieds de tableau (DEFINE FOOTER), des caractéristiques de colonnes (DEFINE COLUMN).

```
PROC TEMPLATE ;
  EDIT nomModeleTabulaire ;
     COLUMN listeDesColonnes ;
     DEFINE HEADER nomEnTete ;
       …
     END ;
     DEFINE FOOTER nomPiedDeTableau ;
```

```
      …
    END ;
    DEFINE COLUMN nomColonne ;
      …
    END ;
  END ;
RUN ;
```

À travers cette redéfinition du modèle tabulaire, on va pouvoir toucher les sorties de toutes les procédures qui l'utilisent. La section 5.2.4 indique comment supprimer un modèle tabulaire personnalisé afin de retrouver l'aspect par défaut des sorties des procédures.

5.2.1 Les en-têtes et pieds de colonnes

Quand on définit un en-tête de colonne, il doit obligatoirement porter un nom, même si ce nom n'est pas mentionné par ailleurs (il ne sert réellement que lorsque l'on définit un en-tête commun à plusieurs modèles tabulaires). Ce nom doit tenir en 32 caractères. La syntaxe sera exactement identique pour la définition d'un en-tête (`DEFINE HEADER`) ou d'un pied de tableau (`DEFINE FOOTER`).

```
DEFINE HEADER|FOOTER nomElement ;
   TEXT "texte"|_LABEL_ ;
 < SPLIT = "caractère" ; >
 < JUST  = LEFT|CENTER|RIGHT ; >
 < VJUST = TOP|MIDDLE|BOTTOM ; >
 < START = nomColonne1 ; >
 < END   = nomColonne2 ; >
END ;
```

On peut indiquer dans l'option `SPLIT` un caractère auquel le texte sera renvoyé à la ligne, comme dans les procédures PRINT et REPORT. Les options `JUST` et `VJUST` indiquent l'alignement de l'en-tête (à gauche, au centre, à droite ; en haut, au milieu, à droite).

L'instruction `TEXT` (sans signe égal !) indique le texte de l'en-tête. On peut y indiquer n'importe quel texte entre guillemets, ou bien le mot-clé `_LABEL_` qui reprend le label de la table SAS. Dans le texte entre guillemets, on peut insérer des instructions de mise en forme après `(*ESC*)` qui remplace le caractère spécial indiqué dans `ODS ESCAPECHAR` (cf. sections 2.2.3 et suivantes). Ainsi, on n'est pas dépendant du choix fait dans l'instruction `ODS ESCAPECHAR` lors de l'utilisation du modèle tabulaire.

Par défaut, les en-têtes sont positionnés au-dessus de l'ensemble des colonnes du tableau. On peut cependant les limiter au-dessus de certaines colonnes avec les options `START` et `END`.

Exemple 5.02 – Ajout d'en-têtes et pieds de tableau à la procédure MEANS

```
PROC TEMPLATE ;  /* partie 1 : définition d'un modèle tabulaire */
  EDIT Base.Summary ; /* objet commun aux procedures Means et Summary */
    COLUMN class nobs varname label mean stddev
           min p1 p5 p10 q1 median q3 p90 p95 p99 max ;
```

```
      DEFINE HEADER quantiles ;
        TEXT "Quantiles de la distribution" ;
        START = min ;
          END = max ;
      END ;
      DEFINE FOOTER note ;
        TEXT "Formule :#(*ESC*){super 1}/(*ESC*){sub (n-1)}
                        (*ESC*)S={font_face='Symbol'}S
                        (*ESC*)S={}(x(*ESC*){sub i}-mean)(*ESC*){super 2}" ;
        START = stddev ;
          END = stddev ;
        SPLIT = "#" ;
      END ;
  END ;
RUN ;
/* partie 2 : utilisation du modèle */
ODS PDF FILE = "c:\temp\template.pdf" ;
PROC MEANS DATA = livre.voitures MEAN STD MIN MEDIAN MAX MAXDEC=2 ;
  VAR puissance ;
  CLASS type ;
RUN ;
ODS PDF CLOSE ;
```

Dans l'exemple 5.02, on ajoute un en-tête qui ne sera affiché qu'au-dessus des quantiles (et donc absent si aucune statistique de ce type n'est réclamée) et une note qui donne la formule (avec force utilisation du caractère d'échappement) de l'écart-type tel que calculé par la procédure. Là encore, le positionnement de ce pied de tableau sous la colonne de l'écart-type exclut son affichage si la statistique en question n'est pas demandée.

Analysis Variable : puissance Puissance réelle (CV vapeur)						
				Quantiles de la distribution		
Type de véhicule	N Obs	Mean	Std Dev	Minimum	Median	Maximum
Berline	11	179.45	21.84	147.00	170.00	214.00
Citadine	21	91.00	21.16	55.00	90.00	140.00
Compacte	16	131.00	22.77	96.00	132.00	172.00
Familiale	22	173.09	52.50	100.00	169.00	295.00
Monospace	9	149.44	19.24	109.00	151.00	170.00
Sportive	14	160.14	74.41	90.00	147.50	300.00
			Formule : $^{1}/_{(n-1)}\Sigma(x_i\text{-mean})^2$			

Figure 5.2 — Résultat de l'exemple 5.02, fichier TEMPLATE.PDF

5.2.2 Les colonnes

On peut redéfinir les différentes colonnes qui composent le modèle tabulaire. On peut ainsi modifier leurs valeurs, leurs formats, leurs en-têtes.

```
DEFINE COLUMN nomColonne ;
  COMPUTE AS formule ;
  TRANSLATE condition1 INTO formule1,
            condition2 INTO formule2,
            …,
            1 INTO formuleN
  ;
  FORMAT = nomFormat. ;
  HEADER = "texte en-tête de colonne" ;
END ;
```

Pour cela, on utilise le bloc `DEFINE COLUMN`, derrière lequel on indique quel est le nom de la colonne modifiée (ce nom se retrouve dans la liste `COLUMN` du modèle tabulaire). Les options `FORMAT` et `HEADER` permettent de modifier les éléments correspondants de la colonne.

L'instruction `COMPUTE AS` fournit une transformation à appliquer systématiquement à chaque valeur (dans l'exemple 5.03, on divise les pourcentages par 100 afin de pouvoir leur appliquer le format approprié). Le mot-clé `_VAL_` désigne la valeur de cette colonne telle qu'elle est renvoyée par la procédure.

Exemple 5.03 – Formule systématique (COMPUTE AS) dans un modèle tabulaire

```
PROC TEMPLATE ;
  EDIT base.freq.oneWayList ;
   DEFINE COLUMN percent ;
      COMPUTE AS _val_/100 ;
      FORMAT = PERCENT12.2 ;
    END ;
    DEFINE COLUMN cumPercent ;
      COMPUTE AS _val_/100 ;
      FORMAT = PERCENT12.2 ;
    END ;
  END ;
RUN ;
PROC FREQ DATA = livre.voitures ;
  TABLE type ;
RUN ;
```

Type de véhicule				
type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Berline	11	11.83%	11	11.83%
Citadine	21	22.58%	32	34.41%
Compacte	16	17.20%	48	51.61%
Familiale	22	23.66%	70	75.27%
Monospace	9	9.68%	79	84.95%
Sportive	14	15.05%	93	100.00%

Figure 5.3 — Résultat de l'exemple 5.03 au format PDF

L'instruction `TRANSLATE INTO` permet de proposer des formules conditionnelles. La première formule (après le premier `INTO`) sera appliquée si la 1re condition est vraie, sinon on évaluera la deuxième condition (et si elle est vraie, on appliquera la deuxième formule), et ainsi de suite jusqu'à la dernière condition (la valeur 1 correspondant à « vrai », il s'agit du cas à appliquer si toutes les conditions précédentes ont été fausses) et à la dernière formule.

Exemple 5.04 – Formule conditionnelle (TRANSLATE) dans un modèle tabulaire

```
PROC TEMPLATE ;
  EDIT base.freq.nLevels ;
    DEFINE COLUMN NMissLevels ;
      TRANSLATE _val_=0 INTO "Aucune obs non renseignée",
                _val_>0 INTO "Au moins 1 obs non renseignée" ;
    END ;
  END ;
RUN ;
PROC FREQ DATA = livre.voitures NLEVELS ;
  TABLE constructeur modele cylindres cylindree vol_coffre / NOPRINT ;
RUN ;
```

Dans l'exemple 5.04, on modifie le modèle tabulaire du tableau `NLEVELS` édité par la procédure FREQ (qui indique le nombre de valeurs distinctes des variables de l'instruction `TABLE` ; il s'agit d'une option apparue avec la version 9 de SAS).

Number of Variable Levels				
Variable	**Label**	**Levels**	**Missing Levels**	**Nonmissing Levels**
constructeur	Constructeur du véhicule	31	Aucune obs non renseignée	31
modele	Nom du modèle	93	Aucune obs non renseignée	93
cylindres	Nombre de cylindres	6	Au moins 1 obs non renseignée	5
cylindree	Cylindrée (L)	26	Aucune obs non renseignée	26
vol_coffre	Volume du coffre (cm3)	17	Au moins 1 obs non renseignée	16

Figure 5.4 — Résultat de l'exemple 5.04 au format.PDF

5.2.3 Les colonnes empilées

L'instruction `COLUMN` permet de lister les éléments du tableau à produire, mais également d'indiquer leur imbrication, comme dans une procédure REPORT. Si on écrit `COLUMN var1 (var2 var3) var4 ;` alors les valeurs des colonnes var2 et var3 seront empilées (sur 2 lignes) dans une même cellule du tableau produit.

L'exemple 5.05 montre comment on peut construire un listing à partir de la procédure MEANS en demandant l'édition d'un intervalle de confiance de la moyenne (mot-clé `CLM`) ; les deux bornes de cet intervalle (LCLM et UCLM) sont déclarées dans `COLUMN`, mais leur définition indique (avec `PRINT=OFF`) qu'elles ne seront pas affichées dans le listing final, au profit de l'information nouvelle IC, qui les regroupe en un seul texte.

Exemple 5.05 – Empilement de statistiques dans une cellule de la proc MEANS

```
PROC TEMPLATE ;
  DELETE base.summary ; /* supprime la version de SASUSER.TEMPLAT */
  EDIT base.summary ;
    COLUMN class nobs (mean lclm uclm ic) ;
    DEFINE COLUMN ic ;
      COMPUTE AS "(entre "!!CATX(" et ",PUT(lclm,12.2),PUT(uclm,12.2))!!")";
    END ;
    DEFINE lclm ;
      PRINT = OFF ;
    END ;
    DEFINE uclm ;
      PRINT = OFF ;
    END ;
  END ;
RUN ;
PROC MEANS DATA = livre.voitures MEAN CLM MAXDEC=2 ;
  VAR conso_auto ;
  CLASS type ;
RUN ;
```

Analysis Variable : conso_auto Consommation sur autoroute (L aux 100 km)		
Type de véhicule	N Obs	Mean
Berline	11	8.82 (entre 8.54 et 9.10)
Citadine	21	6.77 (entre 6.34 et 7.21)
Compacte	16	7.94

Figure 5.5 — Résultat de l'exemple 5.05 au format PDF

5.2.4 La suppression d'un modèle tabulaire personnalisé

Une fois définis, les modèles tabulaires personnalisés seront *systématiquement* utilisés par la procédure correspondante. Afin d'éviter que toutes les sorties d'une procédure donnée soient affichées avec un modèle non standard, on peut adopter deux tactiques.

- soit changer les chemins d'`ODS PATH` (cf. section 5.1.2) afin de stocker le modèle modifié dans un items store à part, et n'y recourir (avec `ODS PATH` encore) que lorsque c'est approprié.
- soit supprimer après emploi le modèle tabulaire personnalisé, avec la proc TEMPLATE et l'instruction `DELETE`. Le modèle sera supprimé du premier items store indiqué dans `ODS PATH` avec les droits `WRITE`.

```
PROC TEMPLATE ;
   DELETE nomModeleTabulaire ;
RUN ;
```

5.2.5 Les modèles tabulaires comme outil de *reporting* ad hoc

Il est possible de construire des modèles tabulaires définis entièrement en fonction d'une table SAS à lister de manière particulière. On substituera alors à la syntaxe `EDIT` un `DEFINE TABLE` qui ne prend aucune définition comme base et demande de préciser chacun des éléments du rapport.

On peut ajouter des éléments paramétrables : des pseudo-variables internes à la procédure TEMPLATE (énumérées après `DYNAMIC` et de type caractère systématiquement, elles prendront leur valeur au moment de l'application du modèle) et des macro-variables (énumérées par leur nom sans `&` après `MVAR` et utilisées ensuite également sans `&` dans le modèle).

```
PROC TEMPLATE ;
  DEFINE TABLE nomModeleTabulaire ;
   < DYNAMIC paramètre1 < paramètre2 < … > > ; >
   < MVAR macro-variable1 < macro-variable2 < … > > ; >
     COLUMN listeDesColonnes ;
     DEFINE HEADER nomEnTete ;
       …
     END ;
     DEFINE FOOTER nomPiedDeTableau ;
       …
     END ;
     DEFINE COLUMN nomColonne ;
       …
     END ;
  END ;
RUN ;
```

Le but d'un tel modèle n'est pas de se cheviller à une procédure, mais d'être invoqués dans une étape DATA, afin de lister une table SAS selon une forme particulière. Par rapport à une procédure PRINT ou REPORT, elle offre des facilités déjà vues pour faire cohabiter en une cellule plusieurs informations, pour associer au tableau créé plusieurs en-têtes de largeur variable, et plusieurs pieds de tableau.

```
DATA _NULL_ ;
   SET tableSAS ;
   FILE PRINT ODS=(TEMPLATE="nomModeleTabulaire"
                < DYNAMIC=(paramètre1="valeur1"
                         < paramètre2="valeur2" … > )
                > ) ;
 < calculs >
   PUT _ODS_ ;
RUN ;
```

L'instruction `FILE` sert d'ordinaire à créer des fichiers plats en sortie de l'étape DATA. `FILE PRINT ODS` est une syntaxe réservée à la liaison entre un modèle tabulaire sur mesure et une table SAS lue dans cette étape Data.

Exemple 5.06 – Modèle tabulaire sur mesure pour le listage d'une table

```
PROC TEMPLATE ;
  DEFINE TABLE listeVoitures ;
    DYNAMIC nb ;
    COLUMN voiture conso_ville conso_auto puissance poids
          (prix_min prix_max) ;
    DEFINE HEADER global ;
      TEXT "Liste de " nb 3. " modèles disponibles aux USA en 1993" ;
    END ;
    DEFINE HEADER conso ;
      START = conso_ville ; END = conso_auto ;
      TEXT "Consommation (L/100 km)" ;
    END ;
    DEFINE COLUMN prix_min ;
      HEADER = "Prix en francs (min et max)" ;
    END ;
```

```
  END ;
RUN ;
DATA _NULL_ ;
  SET livre.voitures NOBS=nbObs ;
  FILE PRINT ODS=(TEMPLATE="listeVoitures" DYNAMIC=(nb=nbObs)) ;
  voiture = CATX(" ",constructeur,modele) ;
  LABEL conso_ville="en ville" conso_auto="sur autoroute" voiture="Modèle" ;
  FORMAT prix_ : NLNUM12. ;
  PUT _ODS_ ;
RUN ;
```

Liste de 93 modèles disponibles aux USA en 1993

Modèle	Consommation (L/100 km)		Puissance réelle (CV vapeur)	Poids du véhicule (tonnes)	Prix en francs (min et max)
	en ville	sur autoroute			
Acura Integra	9.41	7.59	140	1.225	77 400 112 800
Acura Legend	13.07	9.41	200	1.613	175 200 232 200
Audi 90	11.76	9.05	172	1.529	155 400 193 800
Audi 100	12.38	9.05	172	1.542	184 800 267 600
BMW 535i	10.69	7.84	208	1.649	142 200 217 200
Buick LeSabre	12.38	8.4	170	1.572	119 400 130 200

Figure 5.6 — Résultat de l'exemple 5.06 au format PDF

5.3 LA PERSONNALISATION DES STYLES

Un modèle tabulaire personnalisé permet de modifier l'aspect d'un objet produit par une procédure, ou le tableau produit via une étape Data. Le modèle de style permet de personnaliser l'aspect de *l'ensemble* du document produit. On définit également ce modèle avec la procédure TEMPLATE.

Comme pour les modèles tabulaires, on aura une instruction `DEFINE STYLE` et son alternative `EDIT`, selon que l'on veut produire un style au nom inédit, ou modifier un style de nom existant. Contrairement aux modèles tabulaires, la définition d'un nou-

veau style avec DEFINE STYLE peut s'appuyer quand même sur une définition existante, avec la notion d'héritage. On étudiera en détail les éléments de style qui font appel aux polices et aux couleurs, car ce sont les points que l'on souhaite le plus souvent personnaliser. Enfin, une liste plus complète des éléments de style donnera des pistes pour construire des styles sur mesure.

```
PROC TEMPLATE ;
   DEFINE STYLE | EDIT styles.nomStyle ;
   …
   END ;
RUN ;
```

On peut lister un style existant avec la syntaxe donnée à l'exemple 5.07.

Exemple 5.07 – Edition de la définition d'un style existant

```
PROC TEMPLATE ;
  SOURCE styles.sansPrinter ;
RUN ;
```

La définition du style s'affiche dans la fenêtre Log.

```
NOTE: Path 'Styles.sansPrinter' is in : SASHELP.TMPLMST.
define style Styles.sansPrinter;
   parent = styles.Printer;
   replace fonts /
      'TitleFont2' = ("Arial, Helvetica",12pt,Bold Italic)
      'TitleFont' = ("Arial, Helvetica",13pt,Bold Italic)
      'StrongFont' = ("Arial, Helvetica",10pt,Bold)
      'EmphasisFont' = ("Arial, Helvetica",10pt,Italic)
      'FixedEmphasisFont' = ("Courier",9pt,Italic)
      'FixedStrongFont' = ("Courier",9pt,Bold)
      'FixedHeadingFont' = ("Courier",9pt,Bold)
      'BatchFixedFont' = ("SAS Monospace, Courier",7pt)
      'FixedFont' = ("Courier",9pt)
      'headingEmphasisFont' = ("Arial, Helvetica",11pt,Bold Italic)
      'headingFont' = ("Arial, Helvetica",11pt,Bold)
      'docFont' = ("Arial, Helvetica",10pt);
   replace GraphFonts /
      'GraphDataFont' = ("Arial",8pt)
      'GraphValueFont' = ("Arial",10pt)
      'GraphLabelFont' = ("Arial",12pt,Bold)
      'GraphFootnoteFont' = ("Arial",12pt,Bold)
      'GraphTitleFont' = ("Arial",14pt,Bold);
   style Table from Output /
      rules = ALL
      cellpadding = 4pt
      cellspacing = 0.5pt
      borderwidth = 0.5pt;
end;
```

5.3.1 Les héritages

La notion fondamentale dans la gestion des styles est celle d'héritage. Cette notion est commune dans la programmation objet, mais plus rare dans le monde SAS. La définition d'une parenté entre styles indique que, dans le style « fils », tout ce qui n'est pas redéfini est repris à l'identique du style « père ».

Parmi les styles proposés par SAS, des filiations sont déjà existantes : par exemple, le style D3D a pour père BEIGE, qui a pour père DEFAULT. Déterminer les valeurs des divers attributs du style peut ainsi s'apparenter à un jeu de pistes, puisque certaines valeurs sont définies dans le style fils, d'autres dans son père, voire dans son grand-père !

```
PROC TEMPLATE ;
   DEFINE STYLE styles.nomStyle ;
      PARENT = styles.existant ;
      REPLACE élément / attributs ;
      STYLE élément FROM élémentParent / attributs ;
   END ;
RUN ;
```

À l'intérieur d'un style, les différents éléments peuvent également être hérités d'un élément plus générique. On écrit alors `STYLE…FROM` au lieu de `REPLACE`, afin de « recycler » toutes les valeurs qui ne sont pas modifiées de l'élément générique. Là où `REPLACE` écrase l'intégralité d'une définition existante, `STYLE` se contente de l'enrichir. Dans un premier temps, tous les éléments modifiés par une instruction `REPLACE` remplacent entièrement ceux du style parent ; puis tous les autres éléments sont repris à l'identique du style parent, et enfin tous les éléments définis par une instruction `STYLE` sont modifiés.

5.3.2 Les listes de polices et de couleurs

Deux éléments de style sont particulièrement fondamentaux : `FONTS` et `COLORS`. Il s'agit dans les deux cas de listes, dans lesquelles des mots-clés repris dans le reste de la définition du style sont associés à des valeurs. La liste `FONTS` définit les polices utilisées dans le style ; la liste `COLORS` définit les couleurs.

Ces listes ne sont pas interprétées pour elles-mêmes dans la suite de la définition du style ; on n'y trouve que des « surnoms » associés à des définitions de polices ou de couleurs. Si vous ajoutez à la liste `FONTS` un élément `VALEURS`, vous pourrez y faire référence avec l'écriture `FONTS(VALEURS)` dans un autre élément du style. *Pour éviter de devoir tout modifier dans la définition, on utilise en général les mêmes surnoms que ceux déjà présents dans le style dont on part.*

```
REPLACE FONTS /
  "mot-clé" = ("police1, police2, …" < , taillePolice > < , BOLD ITALIC >)
;
REPLACE COLORS /
  "mot-clé" = CXrrggbb|nomCouleurSAS|GRAYhh
;
```

On y utilise obligatoirement le mot-clé `REPLACE`, de manière à éviter que les modifications ne soient prises en compte trop tard (à l'étape 3 de la résolution). La contrepartie de l'emploi de `REPLACE` est qu'il faut énumérer l'ensemble des polices définies dans le style parent, sous peine de laisser des vides qui généreront des messages d'erreur. Par exemple, si le style parent définit trois couleurs appelées BLANC, NOIR et ROUGE, il faut les redéfinir, sous peine qu'une référence à l'élément ROUGE ne trouve plus la définition associée à ce mot-clé.

Cette syntaxe, valable pour les versions 8, 9.1 et 9.2, est simplifiée en version 9.2 avec des écritures alternatives. Comme `REPLACE` fonctionne de manière certaine avec toutes les versions de SAS, c'est la seule que nous détaillerons[1].

Dans la définition d'une police, après un des mots-clés du tableau 5.1, on indique d'abord le nom d'une police entre guillemets, et éventuellement d'autres après une virgule (mais toujours dans la même paire de guillemets) : si la première police n'est pas installée sur le système de celui qui lit le document, la deuxième sera utilisée, et ainsi de suite (il est conseillé de finir la liste avec une police courante, comme Arial, Times ou Courier). Après une virgule, on peut encore préciser la taille[2], souvent sous la forme `10pt` pour indiquer une police de taille 10 comme sous Word. Enfin, après une dernière virgule, on indiquera que la police est en caractères gras (mot-clé `BOLD`) et/ou en italique (`ITALIC`). Si les deux mots-clés sont présents, ils sont séparés par un simple espace.

Tableau 5.1 — Principaux éléments de la liste FONTS

Mot-clé	Usage
TITLEFONT	Texte des instructions TITLE1 à TITLE10
TITLEFONT2	Texte du label de procédure (The xxx Procedure ou ODS PROCLABEL)
HEADINGFONT	En-têtes de lignes et de colonnes
DOCFONT	Texte dans les cellules des tableaux
FOOTFONT	Texte des instructions FOOTNOTE1 à FOOTNOTE10
FIXEDFONT	Texte des sorties en police à pas fixe (type PROC FORMAT FMTLIB, également les notes insérées avec ODS TEXT) non associées à un modèle tabulaire

Dans la définition d'une couleur, les éléments sont résumés dans le tableau 5.2. Les mots-clés se terminent par `FG` ou `BG`, selon qu'il s'agit de la couleur de la police

1. On se reportera à la présentation (disponible en PDF sur Internet) de Kevin D. Smith intitulée *The TEMPLATE Procedure Styles: Evolution and Revolution* et présentée au SUGI 31, pour plus d'informations sur la syntaxe simplifiée proposée dans la version 9.2.
2. Un bug de la version 8.2 fait que lorsqu'on indique une taille de 10PT, elle est rendue en 9,5 dans le document RTF. Ce bug est corrigé en version 9.1, et peut être contourné en indiquant une taille de 10.1PT dans la définition de la police pour obtenir une police de taille 10 points en RTF.

(FG pour FOREGROUND) ou de la couleur du fond de la cellule (BG pour BACKGROUND).On peut leur associer comme valeur soit un mot-clé de couleur SAS, soit un code RGB après les lettres CX, soit une nuance de gris avec un code hexadécimal de GRAY00 (noir) à GRAYFF (blanc). On retrouvera ces codes couleurs détaillés dans la section 2.2.1.

Tableau 5.2 — Principaux éléments de la liste COLORS

Mot-clé	Usage
TITLEFG / TITLEBG	Couleur du texte / du fond des cellules de titres
HEADERFG / HEADERBG	Couleur des en-têtes de colonnes
DATAFG / DATABG	Couleur des cellules de tableaux
BATCHFG / BATCHBG	Couleur des sorties en police à pas fixe
NOTEFG / NOTEBG	Couleur des notes (insérées par PROC DOCUMENT ou ODS TEXT)
TABLEBORDER	Couleur de la bordure du tableau
TABLEBG	Couleur du fond du tableau
BYLINEFG / BYLINEBG	Couleur des lignes générées par les instructions BY
PROCTITLEFG / PROCTITLEBG	Couleur du label de procédure

Exemple 5.08 – Modification des polices d'un style

```
PROC TEMPLATE ;
   DEFINE STYLE styles.dunod ;
      PARENT = styles.journal ;
      REPLACE FONTS /
        "SASTitleFont" = ("Bauhaus 93",12pt,Bold)
        "TitleFont2" = ("Bauhaus 93",12pt,Bold)
        "TitleFont" = ("Bauhaus 93",12pt,Bold)
        "StrongFont" = ("Bauhaus 93",10pt,Bold)
        "EmphasisFont" = ("Bauhaus 93",10pt,Bold Italic)
        "FixedEmphasisFont" = ("Courier New",8PT,Bold Italic)
        "FixedStrongFont" = ("Courier New",8PT,Bold)
        "FixedHeadingFont" = ("Courier New",10PT,Bold)
        "FixedFont" = ("Courier New",8PT)
        "headingEmphasisFont" = ("Bauhaus 93",10pt,Bold Italic)
        "headingFont" = ("Bauhaus 93",10pt,Bold)
        "docFont" = ("Bauhaus 93",10pt)
      ;
   END ;
RUN ;
ODS NOPROCTITLE ;
ODS PDF FILE="c:\temp\styles.pdf" STYLE=JOURNAL COLUMNS=2 ;
TITLE1 J=L "Style journal" J=R "Style personnalisé" ;
```

```
PROC FREQ DATA = livre.voitures ;
       TABLE type ;
RUN ;
ODS PDF STYLE=dunod ;
PROC FREQ DATA = livre.voitures ;
       TABLE type ;
RUN ;
ODS PDF CLOSE ;
```

Style journal

Type de véhicule

type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Berline	11	11.83	11	11.83
Citadine	21	22.58	32	34.41
Compacte	16	17.20	48	51.61
Familiale	22	23.66	70	75.27
Monospace	9	9.68	79	84.95
Sportive	14	15.05	93	100.00

Style personnalisé

Type de véhicule

type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Berline	11	11.83	11	11.83
Citadine	21	22.58	32	34.41
Compacte	16	17.20	48	51.61
Familiale	22	23.66	70	75.27
Monospace	9	9.68	79	84.95
Sportive	14	15.05	93	100.00

Figure 5.7 — Résultat de l'exemple 5.08, fichier STYLES.PDF

5.3.3 Les éléments-clés d'un style

Il existe plus d'une centaine d'éléments de style, dont certains ne sont associés qu'à certaines destinations de l'ODS. Les éléments listés ci-après sont ceux sur lesquels on souhaite le plus souvent pouvoir agir.

```
REPLACE élément FROM élément
   attributs
;
```

Ces éléments ont entre eux des formes d'héritages de leurs propriétés, comme il en existe entre les styles. Le tableau 5.3 reprend les éléments de style, avec leur portée (les destinations ODS qui en tiennent compte, parmi PDF, RTF, HTML et MARKUP/TAGSETS) et les filiations dont ils font l'objet. Les principaux attributs susceptibles d'être associés à l'un ou l'autre de ces éléments sont résumés dans le tableau 5.4.

Tableau 5.3 — Principaux éléments d'un style

Élément	Usage	Parent	Destinations
CONTAINER	Abstrait : ne sert que de parenté aux autres		toutes
DOCUMENT	Abstrait : toutes les sorties	CONTAINER	toutes

Élément	Usage	Parent	Destinations
BODY	Corps du document	DOCUMENT	toutes
CONTENTS	Table des matières	DOCUMENT	toutes sauf RTF
BODYDATE	Date dans le corps du document	DATE	PDF, RTF
SYSTEMTITLE, SYSTEMFOOTER	Titre (TITLE1…TITLE10), pied de page (FOOTNOTE1…10)	TITLESANDFOOTERS	toutes
PAGENO	Numéro de page	TITLESANDFOOTERS	PDF, RTF
BYLINE	Séparations de blocs BY	TITLESANDFOOTERS	toutes
PROCTITLE	Intitulé du nom de procédure	TITLESANDFOOTERS	toutes
TABLE	Toute sortie organisée en tableau	OUTPUT	toutes
BATCH	Toute sortie en police à pas fixe	OUTPUT	toutes
DATA	Contenu d'une cellule	CELL	toutes
HEADER	Cellule d'en-tête	HEADERSANDFOOTERS	toutes
ROWHEADER	Numéro de ligne / identifiant	HEADER	toutes

Tableau 5.4 — Principaux attributs d'un élément de style

Attribut	Valeurs possibles	Usage
LEFTMARGIN, RIGHTMARGIN, TOPMARGIN, BOTTOMMARGIN	nombre suivi de MM, CM ou IN	Marges du document (attention à l'option PAPERSIZE pour les dimensions de la page à proprement parler) en millimètres, centimètres ou pouces
CELLHEIGHT, CELLWIDTH	nombre suivi de MM ou IN	Taille de la cellule (hauteur, largeur) en millimètres ou en pouces
CELLSPACING	nombre suivi de MM ou IN	Espacement entre deux cellules (s'il est supérieur à zéro, on voit la couleur de fond du tableau)
CELLPADDING	nombre suivi de MM ou IN	Espace minimum séparant le texte d'une cellule de ses bordures
BORDERWIDTH	nombre suivi de MM ou IN	Largeur des bords de la cellule

Attribut	Valeurs possibles	Usage
OUTPUTWIDTH	nombre suivi de MM, IN ou %	Largeur du tableau en millimètres, pouces ou en pourcentage de la largeur de page
BACKGROUND	LISTE("alias")	Couleur de fond
FOREGROUND	LISTE("alias")	Couleur de la police
BORDERCOLOR	LISTE("alias")	Couleur des bords de cellule
FRAME	VOID, HSIDES, VSIDES, BOX, ABOVE, BELOW, LHS, RHS	Bordures autour du tableau (VOID=aucune, HSIDES=haut et bas, VSIDES=gauche et droite, BOX=toutes, ABOVE=haut, BELOW=bas, LHS=gauche, RHS=droit)
RULES	ALL, COLS, ROWS, GROUPS, NONE	Bordures à l'intérieur d'un tableau (ALL=toutes, COLS=verticales, ROWS=horizontales, GROUPS=séparation entre en-tête et tableau, NONE=aucune)
FONT	LISTE("alias")	Police utilisée
ASIS	ON, OFF	Affichage des blancs situés à la gauche du texte (ASIS=ON) ; par défaut, ASIS=OFF
FLYOVER	texte	Info-bulle (HTML) ou note (PDF) ; rien en RTF
URL	adresse de fichier (relative ou absolue)	Lien hypertexte
JUST	LEFT, RIGHT, CENTER	Justification horizontale
VJUST	TOP, BOTTOM, MIDDLE	Justification verticale
BACKGROUNDIMAGE	adresse d'un fichier image GIF ou JPEG	Image de fond pour un tableau ou un graphique
PREIMAGE ou POSTIMAGE	adresse d'un fichier image GIF ou JPEG (relative de préférence en HTML, absolue en RTF et PDF)	Insertion d'une image avant ou après le texte

Exemple 5.09 – Style personnalisé

```
OPTION NODATE NONUMBER NOCENTER ;
TITLE ;
PROC TEMPLATE ;
  DEFINE STYLE styles.dunod ;
    PARENT = styles.journal ;

    REPLACE color_list /
      "gris" = "GRAYA0"
      "grisFoncé" = "GRAY50"
    ;
    REPLACE FONTS /
        "SASTitleFont" = ("Century, Tahoma",12pt,Bold)
        "TitleFont2" = ("Century, Tahoma",12pt,Bold)
        "TitleFont" = ("Century, Tahoma",12pt,Bold)
        "StrongFont" = ("Century, Tahoma",10pt,Bold)
        "EmphasisFont" = ("Century, Tahoma",10pt,Bold Italic)
        "FixedEmphasisFont" = ("Courier New",8PT,Bold Italic)
        "FixedStrongFont" = ("Courier New",8PT,Bold)
        "FixedHeadingFont" = ("Courier New",10PT,Bold)
        "FixedFont" = ("Courier New",8PT)
        "headingEmphasisFont" = ("Century, Tahoma",10pt,Bold Italic)
        "headingFont" = ("Century, Tahoma",10pt,Bold)
        "docFont" = ("Century, Tahoma",10pt)
    ;
    STYLE table FROM table /
      RULES = GROUPS
      BORDERWIDTH = 0.1MM
      BORDERCOLOR = color_list("grisFoncé")
      FRAME = BOX
    ;
    STYLE header FROM header /
      BACKGROUND = color_list("gris")
    ;
  END ;
RUN ;
ODS PDF FILE = "c:\temp\style.pdf" STYLE = dunod ;
PROC MEANS DATA = livre.voitures MEAN MEDIAN MAXDEC=1 ;
       VAR conso_ville ;
       CLASS type ;
RUN ;
ODS PDF CLOSE ;
```

Dans l'exemple 5.09, on dérive du style JOURNAL un style DUNOD qui a ses en-têtes de tableau sur fond gris clair, ses bordures gris foncé, horizontales uniquement, dont les polices sont Century, ou à défaut Tahoma.

Analysis Variable : conso_ville Consommation en ville (L aux 100 km)			
Type de véhicule	N Obs	Mean	Median
Berline	11	12.9	12.4
Citadine	21	8.2	8.1
Compacte	16	10.4	10.2
Familiale	22	12.1	12.4
Monospace	9	13.9	13.8
Sportive	14	11.1	10.5

Figure 5.8 — Résultat de l'exemple 5.09, fichier STYLE.PDF

5.3.4 Les références aux éléments d'un style dans une procédure

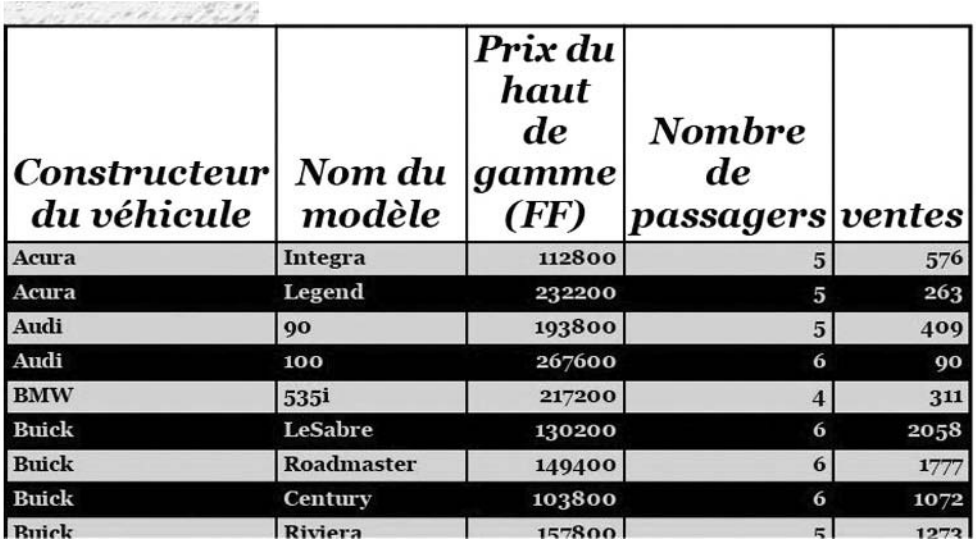
Il est également possible d'utiliser les attributs d'un style (en particulier les éléments des listes de couleurs et de polices) dans les procédures PRINT, REPORT et TABULATE. Cependant, on utilisera cette possibilité avec parcimonie, les alias utilisés dans les listes de couleurs et de polices n'étant pas universels : ils varient (parfois fortement) d'un style à l'autre, même dans une filiation.

L'exemple 5.10 reprend le fonctionnement de l'exemple 3.39 sur les lignes de couleurs alternées en utilisant cette fois les couleurs de fond et de police indiquées dans le style, sans les nommer explicitement. Cela ne suffit malheureusement pas à rendre le fonctionnement du programme universel, puisqu'il faut que les couleurs appelées COLORS(DATAFG) et COLORS(DATABG) soient définies dans le style global.

Exemple 5.10 – Utiliser les éléments du style global dans une procédure.

```
ODS PDF FILE="c:\temp\couleurs.pdf" STYLE=SKETCH ;
PROC REPORT DATA = livre.voitures NOWD ;
  COLUMNS constructeur modele prix_max passagers ventes ;
  DEFINE constructeur / DISPLAY ;
  DEFINE modele / DISPLAY ;
  DEFINE prix_max / DISPLAY ;
  DEFINE passagers / DISPLAY ;
  DEFINE ventes / DISPLAY ;
  COMPUTE BEFORE ;
    numLigne = 0 ;
  ENDCOMP ;
  COMPUTE constructeur ;
    numLigne + 1 ;
    IF MOD(numLigne,2)=0 THEN
         CALL DEFINE(_ROW_,"STYLE",
```

```
        "STYLE=[BACKGROUND=COLORS('datafg') FOREGROUND=COLORS('databg')]") ;
  ENDCOMP ;
RUN ;
ODS PDF CLOSE ;
```



Constructeur du véhicule	Nom du modèle	Prix du haut de gamme (FF)	Nombre de passagers	ventes
Acura	Integra	112800	5	576
Acura	Legend	232200	5	263
Audi	90	193800	5	409
Audi	100	267600	6	90
BMW	535i	217200	4	311
Buick	LeSabre	130200	6	2058
Buick	Roadmaster	149400	6	1777
Buick	Century	103800	6	1072
Buick	Riviera	157800	5	1273

Figure 5.9 — Résultat de l'exemple 5.102, fichier COULEURS.PDF

6

La gestion *a posteriori* des sorties

Objectifs

Ce chapitre décrit l'utilisation de l'ODS DOCUMENT et de la procédure DOCUMENT qui lui est associée. Ensemble, ils permettent le stockage d'une sortie SAS au moment de sa production, pour ensuite en réordonner les composants, en modifier les titres et autres mises en forme, pour les réinjecter dans un document ODS. On peut ainsi s'affranchir de l'ordre imposé par les procédures SAS pour l'édition de leurs sorties, inclure ou exclure certains éléments et ajouter des éléments de texte pour enrichir le document final.

6.1 LA NOTION DE DOCUMENT ODS

Le chapitre 1 explique comment les objets ODS sont au cœur de la transcription de sorties SAS en documents extérieurs ; ils portent l'information produite par la procédure jusqu'à l'ODS qui se charge de sa mise en forme. Le circuit « direct » (de la procédure directement dans le fichier produit par l'ODS) peut être doublé ou remplacé par un circuit indirect, qui pointe vers un *document ODS*. On y stocke les objets ODS au fur et à mesure de leur création.

Lors de leur stockage, les objets produits par la procédure sont « congelés », mis en attente. Leur contenu est figé, leur mise en forme suspendue. On pourra ensuite les réordonner, les sélectionner, modifier leurs titres, pieds de pages, etc.

Le stockage est assuré par une destination ODS particulière, appelée DOCUMENT. Le traitement a posteriori du contenu du document ODS se fait à l'aide de la procédure DOCUMENT.

6.1.1 La création d'un document ODS

DOCUMENT est une destination ODS ; la seule différence que l'on peut noter par rapport aux autres destinations (HTML, RTF, PDF) est qu'on n'a pas d'option `FILE` pour indiquer l'emplacement de stockage et que les notions de mise en forme (attributs de style) en sont absentes. Un document est stocké dans un fichier appelé *items store* ; ce type de fichier est considéré par SAS comme contenant des arborescences, mais pour le système d'exploitation il n'y a qu'un seul fichier (d'extension SAS7BITM sous Windows). On retrouvera la notion d'items store pour les modèles décrits au chapitre 5.

```
ODS DOCUMENT NAME = bib.nomItemStore < (UPDATE|WRITE) > ;
PROC xxx … ;
…
RUN ;
…
ODS DOCUMENT CLOSE ;
```

Dans l'instruction `ODS DOCUMENT` ouvrante, on indique l'items store de stockage dans l'option `NAME`. Comme un nom de table SAS ou de catalogue, il est précédé d'un nom de bibliothèque. À côté du nom de l'items store, on peut indiquer les options `(WRITE)` ou `(UPDATE)`. `(WRITE)` spécifie qu'on écrasera un éventuel document du même nom, tandis qu'avec `(UPDATE)` on ajoutera à un document existant. La valeur par défaut est `(UPDATE)`. Attention, avec `UPDATE`, on ajoute des éléments au *début* du document.

Exemple 6.01 – Création d'un document

```
ODS DOCUMENT NAME=sasuser.exemple (WRITE) ;
  PROC PRINT DATA = livre.voitures NOOBS LABEL ;
    BY type ;
  RUN ;
  PROC FREQ DATA = livre.voitures ;
    TABLE type ;
  RUN ;
ODS DOCUMENT CLOSE ;
```

6.1.2 La visualisation d'un document ODS

Les items stores ne sont pas visibles dans la fenêtre Explorer de SAS. On peut uniquement visualiser le contenu des documents sauvegardés *via* une fenêtre Documents (pour l'afficher, taper `ODSDOC` dans la fenêtre de commande). Le document SASUSER.EXEMPLE créé par l'exemple 6.01 est alors visible dans tout son détail.

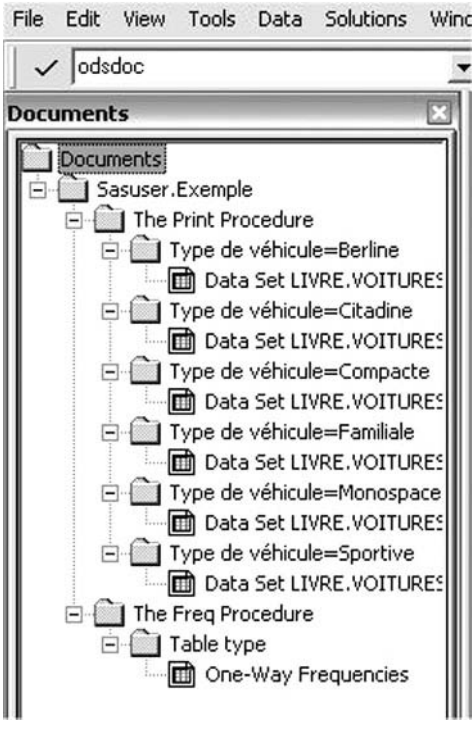


Figure 6.1 — La fenêtre Documents après exécution de l'exemple 6.01

À noter qu'on peut utiliser cette fenêtre, la souris et un menu contextuel (clic droit) pour copier, déplacer, supprimer des sorties de manière interactive. Cette utilisation n'est pas davantage décrite dans les pages qui suivent, qui se limiteront à réaliser ces opérations par du code dans la procédure DOCUMENT.

6.1.3 Que contient un document ODS ?

Un document ODS est composé des objets ODS créés par les différentes procédures qui l'ont alimenté. Les titres et pieds de page en vigueur au moment de la production du document sont également inclus dans ce document.

Les objets ODS du document sont déjà mis en forme par un modèle tabulaire, mais pas encore pas un modèle de style (la notion de modèle, ou *template*, est détaillée dans le chapitre 5). S'il s'agit d'un graphique, il est inscrit dans le document après production *via* un driver qui n'est plus modifiable (la notion de driver graphique est expliquée au début du chapitre 4).

On peut dire que le format du document est neutre, au sens où il n'est pas encore préparé pour une destination ODS en particulier.

Si la finalité première d'un document reste bien de permettre d'effectuer des traitements a posteriori sur les sorties d'une procédure, on peut également considérer que c'est un moyen de stockage des sorties produites à un instant T.

Dans le document, l'organisation est arborescente, comme on peut le constater dans la fenêtre Documents. Une procédure constitue le 1er niveau, puis on retrouve un niveau par bloc BY éventuel, puis les niveaux usuels de la procédure, identiques à ceux vus dans la fenêtre Results.

6.1.4 Les problèmes avec les procédures PRINT et REPORT

À l'exécution de l'exemple 6.01, un message s'affiche dans la fenêtre Log, qui met en garde contre la compatibilité incomplète entre la procédure PRINT et la destination DOCUMENT.

```
WARNING: PROC PRINT does not fully support the ODS Document in this release.
```

La fenêtre Documents indique que l'on a stocké autant d'éléments que de types de véhicules, ce qui semble cohérent avec l'instruction `BY type ;` incluse dans la procédure PRINT. En réalité, ici, la notion de BY est ignorée, et l'arborescence est peuplée de 6 listings intégraux de la table !

Dans les versions 9.1 et 9.2 de SAS, de nombreux dysfonctionnements entachent la relation entre la procédure PRINT et ODS DOCUMENT. Quant à la procédure REPORT, elle ne fonctionne pas avec ODS DOCUMENT en version 9.1, mais semble fonctionner sans erreurs en 9.2 (cependant, la documentation de SAS indique toujours l'incompatiblité entre proc REPORT et ODS DOCUMENT). *Il faut donc éviter d'utiliser ces deux procédures conjointement avec ODS DOCUMENT.*

Les alternatives possibles sont de se tourner vers la procédure TEMPLATE (cf. section 3.2), une personnalisation d'une autre procédure avec un modèle tabulaire (cf. section 5.2), ou une étape `DATA _NULL_` utilisant un modèle tabulaire *ad hoc* pour lister une table (cf. section 5.2.5). Toutes sont compatibles avec ODS DOCUMENT.

6.2 LE POST-TRAITEMENT AVEC LA PROCÉDURE DOCUMENT

La procédure DOCUMENT permet de réaliser de nombreuses opérations sur un document ODS, après production des sorties. En particulier, elle permet de rejouer à la demande les composants du document pour les envoyer dans une destination ODS (hors ODS OUTPUT), mais également de les réordonner, d'en masquer ou d'en supprimer certains, de modifier les titres et les pieds de page, d'insérer des sous-titres et des annotations dans la sortie ODS produite.

```
PROC DOCUMENT NAME = bib.nomItemStore ;
   LIST < arborescence > / LEVELS = ALL ;
 < opérations sur le contenu du document >
   REPLAY arborescence\^ / LEVELS= ALL ;
   REPLAY cheminCompletD'unObjet ;
RUN ; QUIT ;
```

On retrouve dans la syntaxe de base de la procédure DOCUMENT l'option NAME déjà vue lors de l'ouverture d'ODS DOCUMENT. À noter que la procédure se termine par un RUN *et* un QUIT. À l'intérieur de cette procédure, commençons par deux manipulations indispensables : LIST et REPLAY.

6.2.1 La structure arborescente d'un document : répertoires et objets

LIST permet d'afficher le contenu du document, à partir d'un certain niveau de l'arborescence ; sans précision de niveau, LIST affichera le contenu de l'ensemble du document.

Exemple 6.02 – Contenu d'un document

```
PROC DOCUMENT NAME=sasuser.exemple ;
  LIST / LEVELS=ALL ;
RUN ; QUIT ;
```

Listing of: \Sasuser.Exemple\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Print#1	Dir
2	\Print#1\ByGroup1#1	Dir
3	\Print#1\ByGroup1#1\Print#1	Report
4	\Print#1\ByGroup2#1	Dir
5	\Print#1\ByGroup2#1\Print#1	Report
6	\Print#1\ByGroup3#1	Dir
7	\Print#1\ByGroup3#1\Print#1	Report
8	\Print#1\ByGroup4#1	Dir
9	\Print#1\ByGroup4#1\Print#1	Report
10	\Print#1\ByGroup5#1	Dir
11	\Print#1\ByGroup5#1\Print#1	Report
12	\Print#1\ByGroup6#1	Dir
13	\Print#1\ByGroup6#1\Print#1	Report
14	\Freq#1	Dir
15	\Freq#1\Table1#1	Dir
16	\Freq#1\Table1#1\OneWayFreqs#1	Table

Figure 6.2 — Résultat de l'exemple 6.02 au format PDF

La sortie de l'exemple 6.02 indique quelle est l'arborescence du document ; par rapport à la présentation de la fenêtre Documents, on constate que les éléments de l'arborescence ne portent pas les mêmes noms : là où la fenêtre Documents montre des labels, l'instruction `LIST` montre les noms des différents éléments.

La colonne TYPE indique s'il s'agit d'un niveau d'arborescence (`DIR`, pour *directory*, soit répertoire en anglais), ou d'un objet – sous ce nom général, on regroupe les tableaux (`TABLE`), les listes (`REPORT`) et les graphiques (`GRAPH`).

6.2.2 Des sorties rejouées à la demande

Quand on connaît ces éléments de l'arborescence, on peut les rejouer à partir de l'instruction `REPLAY`. Elle permet de relancer l'édition d'une partie de l'arborescence (le niveau « racine » est noté `^`), ou d'un objet particulier, vers les destinations ODS ouvertes.

Exemple 6.03 – Rejouer un document dans un ordre différent

```
ODS PDF FILE = "c:\temp\mon document.pdf" CONTENTS ;
PROC DOCUMENT NAME=sasuser.exemple ;
  REPLAY \Freq#1\^  / LEVELS=ALL ;
  REPLAY \Print#1\^ / LEVELS=ALL ;
RUN ; QUIT ;
ODS PDF CLOSE ;
```

La table des matières du document PDF créé par l'exemple 6.03 montre que les sorties de la procédure FREQ précèdent désormais celles de la procédure PRINT. Les numéros de pages indiquent bien le comportement problématique du couple PRINT / DOCUMENT puisque chaque bloc `BY` fait l'objet de neuf pages de sorties, ce qui est la longueur du listing de la table en entier.

Table of Contents

Figure 6.3 — Résultat de l'exemple 6.03, fichier MON DOCUMENT.PDF

6.2.3 La mise en forme *a posteriori*

Les titres et pieds de page

Il est possible de remplacer les titres et pieds de page qui étaient en vigueur au moment de l'ajout des sorties au document : on dispose pour cela d'instructions `OBTITLE` (numérotées de 1 à 10) et `OBFOOTNOTE` (de même), dans lesquelles on indique l'arborescence et le nom de l'objet dont on veut modifier les titres/pieds de page, puis le texte de remplacement.

```
PROC DOCUMENT NAME = bib.nomItemStore ;
  OBTITLE1 nomObjet "titre" ;
  OBFOOTNOTE1 nomObjet "pied de page" ;
RUN ; QUIT ;
```

Les nouveaux titres et pieds de page sont sauvegardés dans le document. En revanche, il faut rejouer l'objet pour pouvoir les visualiser : ni la fenêtre Documents, ni la proc DOCUMENT ne permettent de savoir quels sont les titres et pieds de page actuellement associés à un objet particulier.

Exemple 6.04 – Modifier les titres *a posteriori*

```
PROC DOCUMENT NAME = sasuser.exemple ;
  OBTITLE1 \Freq#1\Table1#1\OneWayFreqs#1 "Répartition par type" ;
  OBTITLE2 \Freq#1\Table1#1\OneWayFreqs#1 "Re-généré par Proc Document" ;
RUN ;
ODS PDF FILE = "c:\temp\mon document.pdf" ;
  REPLAY \Freq#1\^ / LEVELS=ALL ;
RUN ; QUIT ;
ODS PDF CLOSE ;
```

Répartition par type
Re-généré par Proc Document

The FREQ Procedure

Type de véhicule				
type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Berline	11	11.83	11	11.83
Citadine	21	22.58	32	34.41
Compacte	16	17.20	48	51.61
Familiale	22	23.66	70	75.27
Monospace	9	9.68	79	84.95
Sportive	14	15.05	93	100.00

Figure 6.4 — Résultat de l'exemple 6.04, fichier MON DOCUMENT.PDF

Les sous-titres

Les sous-titres ne sont pas des notions usuelles en SAS. Il y a dix niveaux, accessibles uniquement à travers la procédure DOCUMENT avec les instructions `OBSTITLE1` à `OBSTITLE10`.

```
PROC DOCUMENT NAME = bib.nomItemStore ;
  OBSTITLE1 cheminEtNomObjet "sous-titre" ;
RUN ; QUIT ;
```

À noter que le premier sous-titre correspond au texte « publicitaire » de la procédure, `The XXX Procedure`, que l'on peut par ailleurs désactiver à la création de l'objet *via* l'instruction `ODS NOPROCTITLE`. Son (presque) équivalent a posteriori est `OBSTITLE1 cheminEtNomObjet " " ;` mais il laisse une ligne vide là où `NOPROCTITLE` supprime complètement la ligne.

Exemple 6.05 – Modifier les sous-titres

```
PROC DOCUMENT NAME = sasuser.exemple ;
 OBSTITLE1 \Freq#1\Table1#1\OneWayFreqs#1 "Fréquences=nombre de modèles";
 OBSTITLE2 \Freq#1\Table1#1\OneWayFreqs#1 "Echantillon du marché américain";
RUN ;
ODS PDF FILE = "c:\temp\mon document.pdf" ;
  REPLAY \Freq#1\^ / LEVELS=ALL ;
RUN ; QUIT ;
ODS PDF CLOSE ;
```

Répartition par type
Re-généré par Proc Document

Fréquences=nombre de modèles
Echantillon du marché américain

Type de véhicule				
type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Berline	11	11.83	11	11.83
Citadine	21	22.58	32	34.41
Compacte	16	17.20	48	51.61

Figure 6.5 — Résultat de l'exemple 6.05, fichier MON DOCUMENT.PDF

Les notes

Les notes sont des textes libres que l'on peut incruster dans un document, avant (instruction `OBBNOTE`) ou après l'objet (`OBANOTE`). Il y a dans les deux cas jusqu'à 10 niveaux de notes qui correspondent aux numérotations (`OBBNOTE1`, `OBBNOTE2`, …, `OBBNOTE10`) des instructions.

```
PROC DOCUMENT NAME = bib.nomItemStore ;
   OBBNOTE1 cheminCompletD'unObjet "note 1 avant l'objet" ;
   OBBNOTE2 cheminCompletD'unObjet "note 2 avant l'objet" ;
   …
   OBANOTE1 cheminCompletD'unObjet "note 1 après l'objet" ;
   OBANOTE2 cheminCompletD'unObjet "note 2 après l'objet" ;
   …
RUN ; QUIT ;
```

Les notes après un objet sont insérées *immédiatement* avant ou après le graphique, la liste ou le tableau. Les options de mise en forme des instructions `TITLE` et `FOOTNOTE` n'étant pas disponibles dans les instructions insérant des notes, on utilisera la mise en forme intégrée d'`ODS ESCAPECHAR`.

Exemple 6.06 – Insérer des notes

```
PROC DOCUMENT NAME = sasuser.exemple ;
  OBANOTE1 \Freq#1\Table1#1\OneWayFreqs#1 "(*ESC*)S={JUST=C}Données 1993" ;
ODS PDF FILE = "c:\temp\mon document.pdf" ;
  REPLAY \Freq#1\^ / LEVELS=ALL ;
RUN ; QUIT ;
ODS PDF CLOSE ;
```

Type de véhicule				
type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Berline	11	11.83	11	11.83
Citadine	21	22.58	32	34.41
Compacte	16	17.20	48	51.61
Familiale	22	23.66	70	75.27
Monospace	9	9.68	79	84.95
Sportive	14	15.05	93	100.00

Données 1993

Figure 6.6 — Résultat de l'exemple 6.06, fichier MON DOCUMENT.PDF

6.2.4 La suppression des éléments existants

On peut à tout moment éliminer tout ou partie de l'arborescence d'un document. Attention, l'instruction `DELETE` élimine tout ce qui se trouve à partir de ce niveau d'arborescence !

```
PROC DOCUMENT NAME = bib.nomItemStore ;
   DELETE chemin ;
RUN ; QUIT ;
```

On va éliminer du document SASUSER.EXEMPLE la procédure PRINT qui a été si mal intégrée au document (pour rappel, elle a ses sorties répétées à l'identique six fois au lieu d'un listing éclaté en six blocs BY).

Exemple 6.07 – Supprimer une partie d'un document

```
PROC DOCUMENT NAME=sasuser.exemple ;
  DELETE \Print#1 ;
RUN ; QUIT ;
```

6.2.5 L'insertion d'éléments supplémentaires

L'instruction IMPORT de la procédure DOCUMENT permet d'ajouter à un document une table SAS. Cela permet de palier dans une certaine mesure l'incompatibilité entre la destination DOCUMENT et la procédure PRINT. Même si on ne peut ici obtenir qu'un listing peu souple de la table, on peut néanmoins produire ce type de sortie en rejouant un document.

L'ajout de la table se fait par rapport à dans un niveau existant de l'arborescence du document (par exemple la racine ^), en indiquant qu'on insère soit en premier dans ce niveau (option FIRST) ou tout à la fin des éléments de ce niveau (LAST).

```
PROC DOCUMENT NAME = bib.nomItemStore ;
   IMPORT DATA=tableSAS TO chemin < / FIRST | LAST > ;
RUN ; QUIT ;
```

Dans l'exemple 6.08, on ajoute une partie de la table VOITURES (restreinte par des options de table) au début du document (rappel : ^ marque la racine de l'arborescence du document).

Exemple 6.08 – Ajouter une table SAS à un document

```
PROC DOCUMENT NAME = sasuser.exemple (UPDATE) ;
       IMPORT DATA=livre.voitures (KEEP=constructeur modele puissance type
                                   WHERE=(type = "Sportive")) TO ^ / FIRST ;
RUN ;
ODS RTF FILE="c:\temp\table et document.doc" ;
       REPLAY ^ / LEVELS=ALL ;
RUN ; QUIT ;
ODS RTF CLOSE ;
```

Obs	Constructeur du véhicule	Nom du modèle	Type de véhicule	Puissance réelle (CV vapeur)
1	Chevrolet	Camaro	Sportive	160.000000
2	Chevrolet	Corvette	Sportive	300.000000
3	Dodge	Stealth	Sportive	300.000000
4	Ford	Mustang	Sportive	105.000000
5	Ford	Probe	Sportive	115.000000
6	Geo	Storm	Sportive	90.000000
7	Honda	Prelude	Sportive	160.000000
8	Hyundai	Scoupe	Sportive	92.000000

Figure 6.7 — Résultat de l'exemple 6.08, fichier TABLE ET DOCUMENT.DOC

On retrouve sur la figure 6.7 une colonne OBS comme dans une procédure PRINT ; en revanche, par rapport aux choix par défaut de PRINT, on a ici les labels en tête de colonne. Les formats affectés aux variables numériques sont systématiquement perdus lors du stockage de la table dans le document.

Une solution plus coûteuse en code serait d'utiliser une étape Data avec le modèle tabulaire *ad hoc* comme décrit dans la section 5.2.5. Mais elle permettrait de mieux contrôler l'emploi de labels, les formats d'affichage, et permettrait même de réordonner les variables, d'empiler des valeurs dans une même cellule, et de gérer des en-têtes et pieds de page personnalisés !

7

La mise en œuvre dans l'architecture SAS BI

Objectifs

Les six premiers chapitres de ce livre proposent de nombreuses fonctionnalités de *reporting* grâce à SAS . Ces fonctionnalités ne sont pas disponibles dans toutes les versions, mais elles ne sont pas non plus toutes utilisables, ou alors pas sans précautions, dans tous les environnements qui font travailler SAS.

L'arrivée de SAS Enterprise Guide (SEG) et son ODS automatique et implicite, puis de l'infrastructure SAS Business Intelligence (SAS BI) et de ses « clients » comme l'Add-In For Microsoft Office font que l'on peut désormais exécuter un programme SAS depuis des environnements très divers. Le but de ce chapitre est de recenser les principales particularités du contexte dans lequel vous êtes amené à travailler.

Pour chaque environnement, une même série de questions sera traitée : comment accéder à diverses informations liées à l'ODS, où créer des fichiers via l'ODS, peut-on automatiser (rendre invisibles et systématiques) les instructions liées à l'ODS, et quelles sont les précautions à prendre en particulier par rapport aux diverses techniques déjà présentées dans les pages précédentes.

7.1 LES DIFFÉRENTS ENVIRONNEMENTS DE SAS BI

La figure 7.1 illustre les différents modes de fonctionnement de SAS dans l'infrastructure SAS BI. Historiquement, les environnements de programmation ont été le client seul et le mode client/serveur entre deux sessions SAS. Il y a également les connexions directes à un serveur (via des émulations comme Extra pour MVS, Hummingbird pour Unix et Citrix pour Windows) qui se rapprochent beaucoup du mode client seul.

À partir de l'année 2000, SAS a proposé son logiciel Enterprise Guide, qui permet de rédiger du code dans un environnement plus assisté (on peut y produire des tableaux et des graphiques par des menus et des interfaces presse-bouton). On peut également y exécuter et y écrire du code SAS comme dans une session classique. SAS EG se connecte à un serveur (SEG ne *contient* pas de session SAS) pour exécuter son code, et ressemble au client/serveur « historique ».

Enfin, la version 9 de SAS a coïncidé avec la diffusion de l'architecture BI (ou Business Intelligence) qui met à disposition de nombreux « clients » (dont Excel, Word, des applications web) des sessions SAS « esclaves » auxquelles on peut demander des tâches pré-programmées, appelées applications stockées (ou *stored processes*).

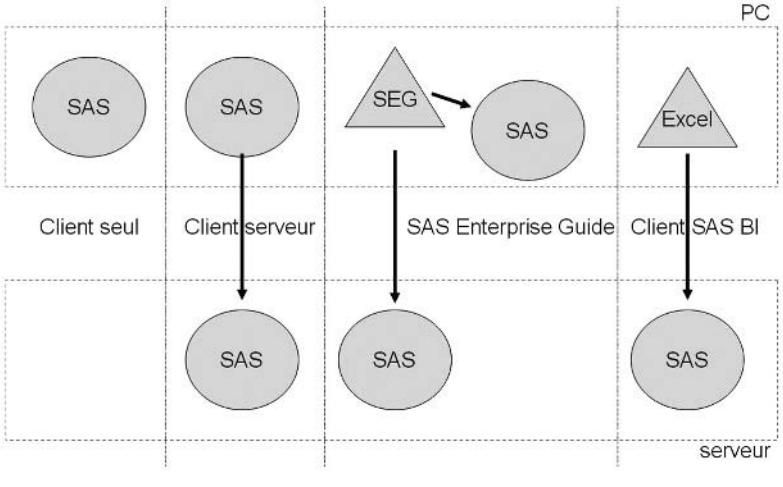


Figure 7.1 — Les différents environnements de SAS BI

7.2 L'ENVIRONNEMENT SAS « CLASSIQUE » EN CLIENT SEUL

Il s'agit de l'installation de SAS sur un PC pouvant fonctionner sans connexion à un réseau. Tous les programmes sont exécutés sur le PC et les sorties sont affichées par défaut dans la fenêtre Output. C'est dans cet environnement qu'ont été écrits et testés tous les programmes de ce livre.

7.2.1 Comment accéder… ?

On peut accéder à la liste des styles ODS, à la liste des modèles tabulaires et à la liste des documents ODS à travers deux fenêtres : Templates et Documents. Ces fenêtres ne sont pas activées par défaut, et on les obtient en exécutant dans la fenêtre de commande (en haut, à gauche de la barre d'outils), la commande `ODSTEMPL` pour la fenêtre Templates, et la commande `ODSDOC` pour la fenêtre Documents. Dans la fenêtre Templates, la liste des styles s'obtient en ouvrant, dans l'*items store* SASHELP.TMPLMST, le répertoire STYLES (figure 7.2).

On peut également ouvrir la fenêtre Templates en se positionnant dans la fenêtre Results, et en faisant un clic droit sur la racine de l'arborescence (RESULTS) : dans le menu contextuel, on choisit TEMPLATES.

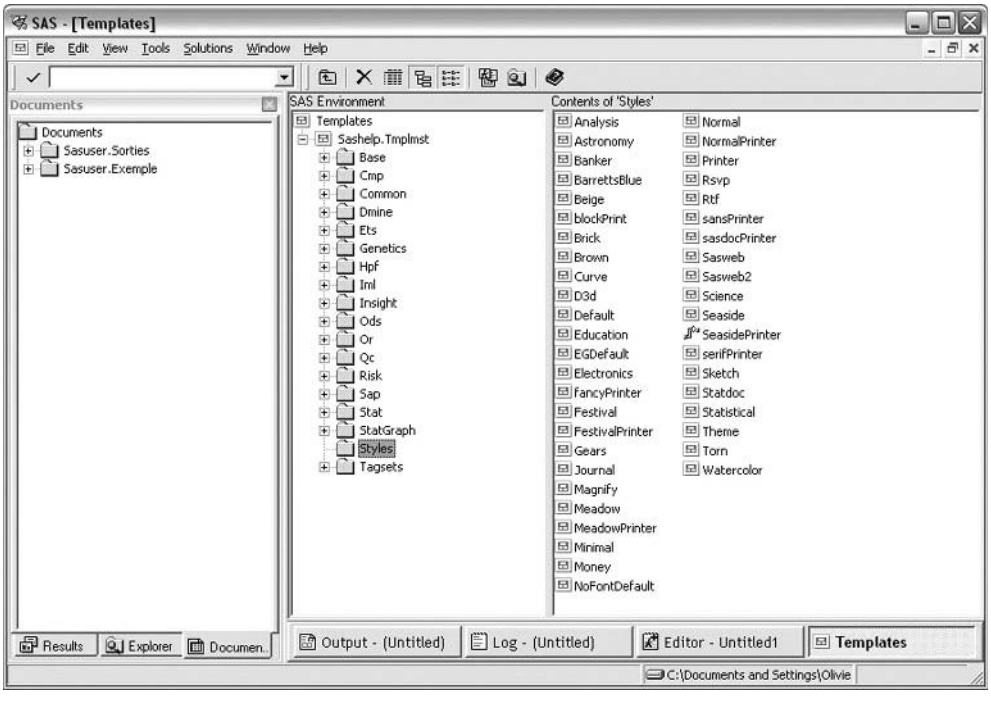


Figure 7.2 — Consulter la liste des styles dans SAS PC (DMS)

Pour consulter les résultats, on utilise la fenêtre Results. Quand l'ODS est actif, chaque objet produit apparaît dans l'arborescence, dupliqué autant de fois qu'il aura été envoyé dans des destinations ODS.

Le menu TOOLS > OPTIONS > PREFERENCES, onglet RESULTS (figure 7.3), permet de contrôler si les fichiers produits par l'ODS sont spontanément ouverts et montrés à l'utilisateur, et avec quel logiciel. La case à cocher VIEW RESULTS AS THEY ARE GENERATED permet de demander, quand elle est active, que tout fichier produit par l'ODS soit affiché quand une instruction `ODS … CLOSE` a été exécutée.

Le bouton radio VIEW RESULTS USING permet de choisir le logiciel utilisé pour afficher ces fichiers : INTERNAL BROWSER désigne l'afficheur universel de SAS (qui affiche indifféremment documents RTF, PDF, HTML, Excel) tandis que PREFERRED WEB BROWSER désigne le logiciel associé par Windows au type de fichier produit (par exemple, Internet Explorer pour les fichiers .htm et Acrobat Reader pour les fichiers .pdf). Il est souvent préférable de choisir un logiciel externe, le navigateur interne de SAS s'avérant assez lent sur de gros fichiers.

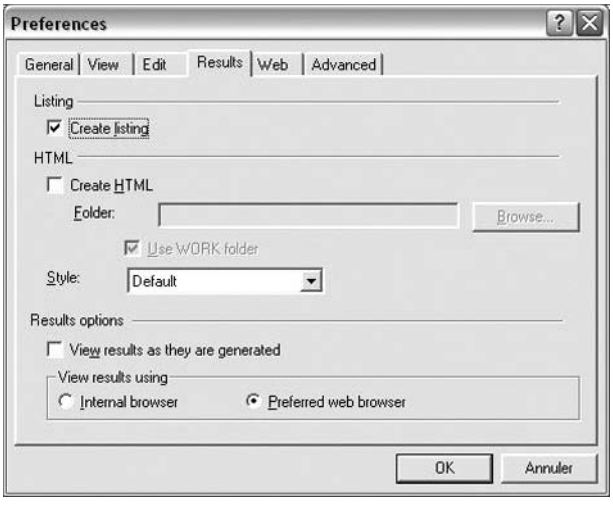


Figure 7.3 — Le menu Tools>Options>Préférences, onglet Results

7.2.2 Où créer des fichiers avec l'ODS ?

Depuis une session SAS sur Windows, on peut créer des fichiers dans tous les répertoires où l'on dispose de droits en écriture, sur les disques locaux comme sur un lecteur réseau. Sur une session SAS ouverte dans un émulateur, on prendra garde à respecter les conventions de noms et d'organisation des fichiers du système hôte.

Pour récupérer en local les fichiers créés en mode émulateur, on utilisera soit un logiciel de transfert FTP, soit un utilitaire de transfert intégré à l'émulateur, soit encore un lecteur réseau *mappé* vers le serveur.

7.2.3 Peut-on automatiser l'ODS ?

Dans le menu TOOLS > OPTIONS > PREFERENCES, onglet RESULTS, déjà vu plus haut, on peut demander l'édition systématique de résultats au format HTML. Cocher pour cela la case CREATE HTML. On peut alors indiquer dans quels répertoires ces fichiers sont stockés (par défaut, la case USE WORK FOLDER est cochée : c'est le répertoire correspondant à la bibliothèque Work, supprimé à la fermeture de SAS, qui contient les fichiers) avec le bouton BROWSE. On peut également imposer que ces fichiers soient produits avec un modèle de style choisi (dans ceux de l'*items store* SASHELP.TMPLMST) avec la liste déroulante STYLE.

On peut également suspendre l'édition de sorties texte dans la fenêtre Output, en décochant la case CREATE LISTING. On a alors l'équivalent de l'instruction `ODS LISTING CLOSE`.

Attention, si les deux cases CREATE LISTING et CREATE HTML sont décochées, en l'absence d'instructions ODS dans le programme, celui-ci ne produira aucune sortie. Un message d'avertissement le signale dans la fenêtre Log.

```
WARNING: No output destinations active.
```

7.2.4 Y a-t-il des précautions à prendre ?

- **Attention aux droits sur les répertoires** pour y écrire des fichiers.
- **Attention au répertoire par défaut !** Si le chemin proposé dans les options `FILE` ou `PATH/GPATH` ne commence pas par une lettre suivie de `:`, alors SAS ajoute systématiquement au début du chemin fourni son répertoire par défaut, défini à l'installation, et par défaut c:\documents and settings*user* où *user* est l'identifiant de l'utilisateur). Par exemple, dans la syntaxe `FILE="c\temp\sortie.htm"` où on aurait oublié les deux points après C, le programme génère une erreur, parce que SAS comprend `FILE="c:\documents and settings\olivier\c\temp\sortie.htm"` et il ne trouve pas de sous-répertoire appelé C. De même une syntaxe `FILE="sortie.htm"` créera bien le fichier, mais le stockera dans ce répertoire par défaut.
- **Attention en mode émulateur** : garder à l'esprit les conventions de noms de fichiers et de répertoires du système.

7.3 L'ENVIRONNEMENT SAS « CLASSIQUE » EN CLIENT/SERVEUR

Depuis un client SAS classique, on se connecte (instruction `SIGNON`) à une session SAS sur un serveur. Le but est généralement d'éviter que les données, stockées sur le serveur, soient copiées en local, et de profiter de la puissance de calcul du serveur.

7.3.1 Comment accéder… ?

Le résultat d'une exécution distante (RSUBMIT) arrive en un seul bloc au client : la notion d'objet ODS ne se retrouve pas en local, en particulier dans la fenêtre Results (comme on le constate sur la figure 7.4).

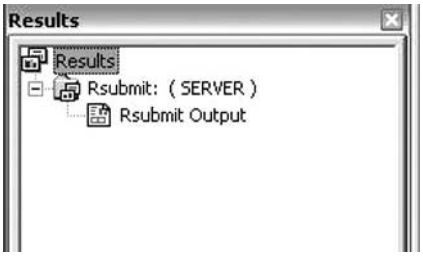


Figure 7.4 — La fenêtre Results après exécution d'un programme via RSUBMIT

Les listes de styles, de documents, de modèles de style situés sur le serveur ne sont pas visibles dans les fenêtres Templates et Documents, même en affectant des bibliothèques RLS. Pour lister le contenu d'un document, on utilisera une procédure DOCUMENT.

```
RSUBMIT ;
PROC DOCUMENT NAME=bibliotheque.document ;
  LIST / LEVELS = ALL ;
RUN ; QUIT ;
ENDRSUBMIT ;
```

Pour lister les styles disponibles, on exécutera une procédure TEMPLATE sur le serveur.

```
RSUBMIT ;
PROC TEMPLATE ;
  LIST styles ;
RUN ;
ENDRSUBMIT ;
```

Pour lister tous les modèles personnalisés, qu'ils soient tabulaires ou de style, on exécutera également une procédure Template. La syntaxe suivante cherche ces modèles dans un *items store* appelé SASUSER.TEMPLAT, mais tout autre items store peut être indiqué après l'option STORE.

```
RSUBMIT ;
PROC TEMPLATE ;
  LIST / STORE=sasuser.templat ;
RUN ;
ENDRSUBMIT ;
```

7.3.2 Où créer des fichiers avec l'ODS ?

Le transfert des sorties SAS du serveur vers le client se fait d'un bloc et pas par objet ODS, on n'inclut pas de traitement distant (RSUBMIT) *à l'intérieur* d'un « sandwich » ODS local. On verra dans la section 7.3.3 avec l'exemple 7.03 qu'une telle organisation du programme conduit à un résultat hideux. Heureusement, il existe deux alternatives : exécuter l'ODS et le traitement de rendu en local, ou tout exécuter sur le serveur.

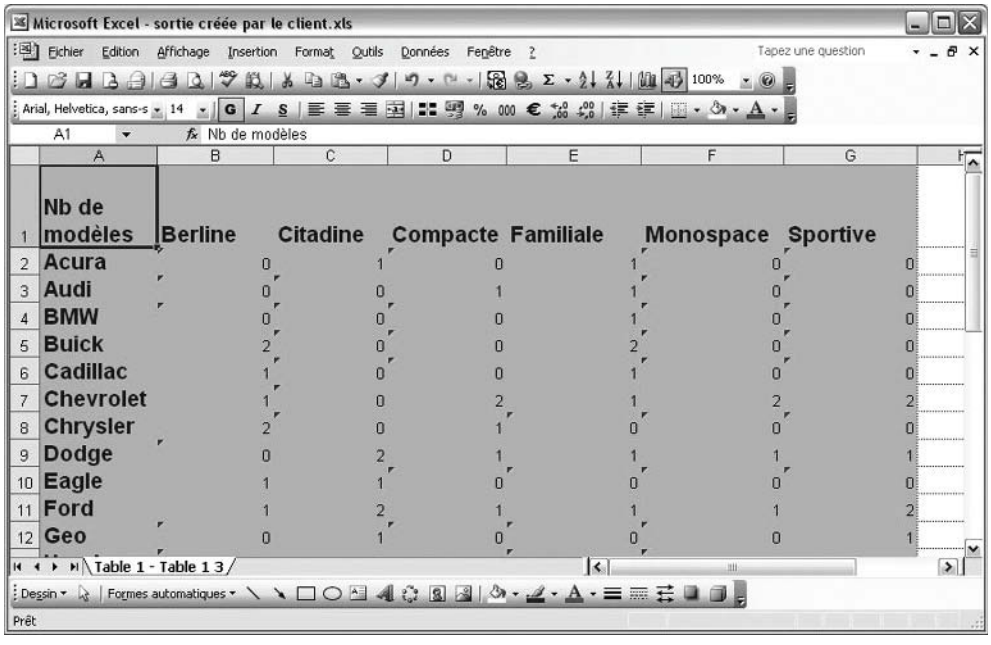
L'ODS en local

On peut exécuter en local les traitements d'affichage (Print, Tabulate, Report, Gchart, Gplot) dans des sandwiches ODS, après avoir descendu les données nécessaires (proc DOWNLOAD ou bibliothèque RLS) du serveur. Pour des raisons de performance, on essaiera au maximum d'effectuer les calculs de statistiques sur le serveur (RSUBMIT), en stockant le résultat dans une table sur le serveur, qu'on descend ensuite. On choisit alors comme emplacement des fichiers ODS tout répertoire dans lequel on a des droits d'écriture, comme dans le mode client seul décrit précédemment.

Cette solution n'est à préférer que dans les cas où on ne pourrait pas construire de sortie équivalente sur le serveur. C'est le cas par exemple avec des graphiques, puisque le module SAS/GRAPH n'est pas toujours installé sur les serveurs, de même que le *driver* ActiveX (disponible uniquement sur un serveur *Windows*).

Exemple 7.01 – Utiliser l'ODS en local

```
RSUBMIT ;
ODS OUTPUT list = work.stats ;
PROC FREQ DATA = livre.voitures ;
  TABLE constructeur * type / LIST NOCUM NOPERCENT ;
RUN ;
PROC DOWNLOAD DATA = work.stats OUT = work.stats_local ;
RUN ;
ENDRSUBMIT ;
ODS TAGSETS.EXCELXP FILE="c:\temp\sortie créée par le client.xls" ;
PROC TABULATE DATA = work.stats_local FORMAT=5. ;
  CLASS constructeur type ;
  VAR frequency ;
  TABLE constructeur="",
        type="" * frequency="" * SUM="" / BOX="Nb de modèles" MISSTEXT="0" ;
RUN ;
ODS TAGSETS.EXCELXP CLOSE ;
```



Nb de modèles	Berline	Citadine	Compacte	Familiale	Monospace	Sportive
Acura	0	1	0	1	0	0
Audi	0	0	1	1	0	0
BMW	0	0	0	1	0	0
Buick	2	0	0	2	0	0
Cadillac	1	0	0	1	0	0
Chevrolet	1	0	2	1	2	2
Chrysler	2	0	1	0	0	0
Dodge	0	2	1	1	1	1
Eagle	1	1	0	0	0	0
Ford	1	2	1	1	1	2
Geo	0	1	0	0	0	1

Figure 7.5 — Résultat de l'exemple 7.01, fichier SORTIE CREEE PAR LE CLIENT.XLS

Dans l'exemple 7.01, la procédure Freq initiale permet de réduire le volume des données à transférer sur le serveur : la table STATS qu'elle crée ne contient qu'une observation par couple de valeurs des variables CONSTRUCTEUR et TYPE, et non plus le détail de toutes les observations initiales. On utilise ensuite les comptages de Freq comme pondération dans le tableau produit en local.

L'ODS sur le serveur

On peut également exécuter la mise en forme et les instructions ODS sur le serveur (dans un bloc `RSUBMIT`) puis transférer en local les fichiers produits. Comme les fichiers sont créés sur le serveur, on doit utiliser un répertoire où les droits en écriture sont accordés à l'utilisateur ; on pourra utiliser, pour cet usage transitoire, le répertoire associé à la bibliothèque WORK, comme le propose l'exemple 7.02.

Le résultat est en tous points semblable à la figure 7.5, à part le nom du fichier.

Exemple 7.02 – Utiliser l'ODS sur le serveur

```
RSUBMIT ;
ODS TAGSETS.EXCELXP FILE="%SYSFUNC(GETOPTION(WORK))/sortie.xls" ;
PROC TABULATE DATA = livre.voitures FORMAT=5. ;
  CLASS constructeur type ;
  TABLE constructeur="",
        type="" * N="" / BOX="Nb de modèles" MISSTEXT="0" ;
RUN ;
ODS TAGSETS.EXCELXP CLOSE ;
```

```
PROC DOWNLOAD INFILE="%SYSFUNC(GETOPTION(WORK))/sortie.xls"
              OUTFILE="c:\temp\sortie créée par le serveur.xls"
               BINARY ;
RUN ;
ENDRSUBMIT ;
```

7.3.3 Peut-on automatiser l'ODS ?

Si on active sur le client l'option pour générer automatiquement une sortie HTML (cf. section 7.2.3), on obtient une transcription par un ODS local du traitement distant, comme si on avait encadré d'instructions `ODS HTML` un bloc `RSUBMIT`.

Le résultat n'est pas esthétique, loin s'en faut.

Exemple 7.03 – L'option CREATE HTML sur une session SAS client/serveur

```
RSUBMIT ;
PROC FREQ DATA = livre.voitures ;
  TABLE type ;
RUN ;
ENDRSUBMIT ;
```

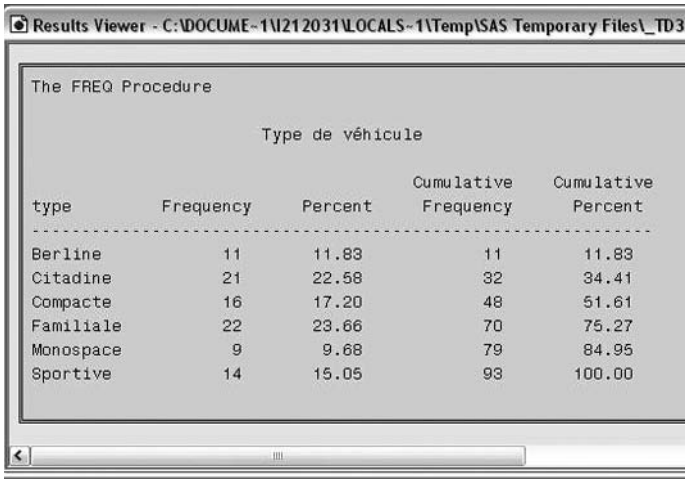


Figure 7.6 — Résultat de l'exemple 7.03 au format HTML

7.3.4 Y a-t-il des précautions à prendre ?

Selon les serveurs, il n'est pas toujours possible d'utiliser les fonctionnalités comme les graphiques Java du langage GTL (cf. chapitre 8) et d'ODS GRAPHICS (cf. section 4.9) ; en version 9.1, ces graphiques ne peuvent être produits que sur des serveurs Windows. La version 9.2 doit corriger cette limitation.

Le driver graphique ACTIVEX et sa variante ACTXIMG (cf. section 4.1) ne peuvent également être utilisés pour produire des graphiques que sur un serveur Windows.

Les documents créés par la destination ODS DOCUMENT (cf. chapitre 6) ne peuvent être utilisés par des sessions SAS sur un système d'exploitation différent. On ne pourra donc créer un document sur le serveur, et le rejouer sur le client, que s'il s'agit d'un serveur Windows et d'un client PC.

7.4 L'ENVIRONNEMENT SAS ENTERPRISE GUIDE

Programmer dans l'environnement SEG (*SAS Enterprise Guide*), quelle que soit la version utilisée (les versions 2.0, 3.0 et 4.1 cohabitaient encore lors de la rédaction de ce livre) est d'autant plus agréable que des instructions ODS implicites permettent d'acheminer automatiquement les résultats vers des fichiers RTF, PDF ou HTML, que l'on fonctionne avec un « serveur local » ou un serveur distant. Il est possible de modifier les options de ces destinations si le serveur est une session de SAS version 9 : on utilisera pour cela des instructions ODS sans option `FILE` pour modifier les choix par défaut.

On peut également ajouter aux programmes ses propres instructions ODS, en les nommant (cf. chapitre 1) pour éviter toute confusion s'il s'agit de destinations RTF, PDF ou HTML, et sans autre précaution pour les autres destinations.

7.4.1 Comment accéder… ?

On ne dispose pas dans *SAS Enterprise Guide* des fenêtres Templates et Documents. Pour lister le contenu d'un document, on utilisera les mêmes programmes que dans la section 7.3.1 sur le mode client/serveur, en omettant les instructions RSUBMIT et ENDRSUBMIT.

7.4.2 Où créer des fichiers avec l'ODS ?

Les précautions sont ici les mêmes qu'en mode client/serveur. Il faut savoir quel est le serveur (local, serveur Windows, serveur Unix ou serveur MVS) pour connaître les normes de nommage d'un fichier et les répertoires dans lesquels on peut écrire. Par prudence, on peut utiliser le répertoire de la bibliothèque WORK avec la même syntaxe qu'à l'exemple 7.02 (`%SYSFUNC(GETOPTION(WORK))` indique le chemin de la WORK).

7.4.3 Peut-on automatiser l'ODS ?

C'est un des grands intérêts de SEG : par rapport à une session SAS classique, le choix de destinations ODS automatiques est plus grand.

Les captures d'écrans présentées ci-après proviennent de la version 4.1 de *SAS Enterprise Guide*.

SAS Enterprise Guide 2.0

Dans le menu OUTILS > OPTIONS… et la rubrique RESULTATS > GENERAL, on trouvera une série de cases à cocher pour activer les différentes destinations ODS de manière systématique. SEG n'affiche dans l'arborescence du projet que les fichiers produits par des instructions ODS *implicites*.

SAS Enterprise Guide 3.0 et 4.1

Dans le menu OUTILS > OPTIONS… et la rubrique RESULTATS > GENERAL, on trouve une série de cases à cocher pour activer les différentes destinations ODS de manière systématique. Par défaut, SEG insère automatiquement dans le projet des résultats au format PDF, RTF et HTML. On peut demander l'insertion des fichiers produits par les instructions ODS *explicites* avec la case à cocher LIER LES RESULTATS ODS CODES MANUELLEMENT.

Dans les versions 3.0 et 4.1 on trouve ensuite, dans les rubriques HTML, RTF et PDF de RESULTATS, des options à ajouter à ces destinations.

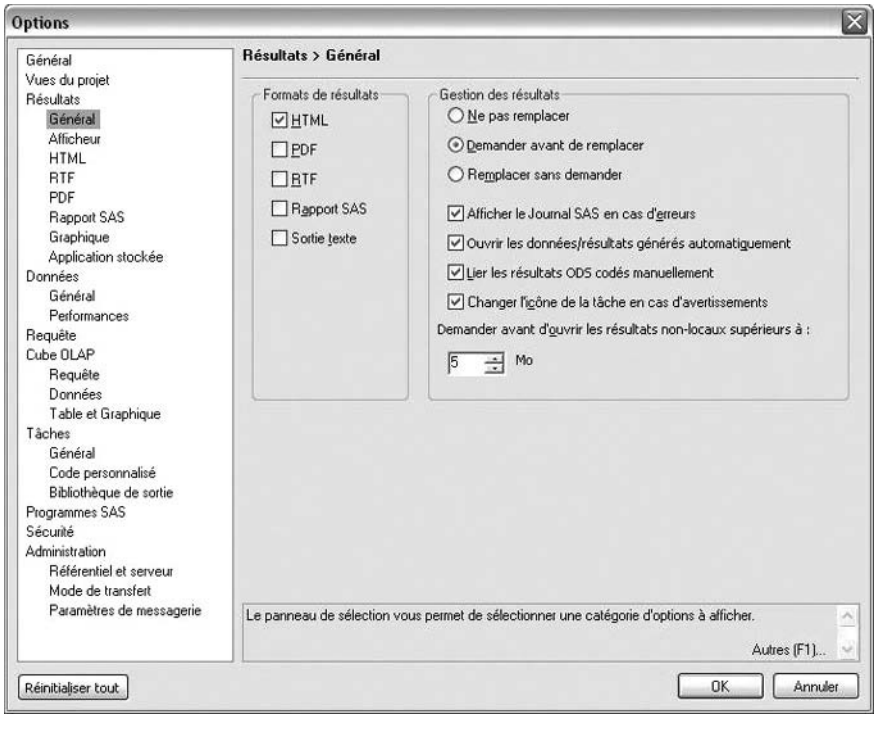


Figure 7.7 — Le menu options dans SAS Enterprise Guide 4.1

La modification des options générales pour un traitement particulier

De plus, dans toutes les versions, on peut réévaluer tâche par tâche et programme par programme lesquelles de ces destinations automatiques seront utilisées. Pour cela, on clique avec le bouton droit sur une tâche ou un code dans le projet et on affiche ses propriétés. Hormis une esthétique variable selon les versions, la logique reste la même. Il est nécessaire d'exécuter à nouveau le traitement pour que les changements soient pris en compte.

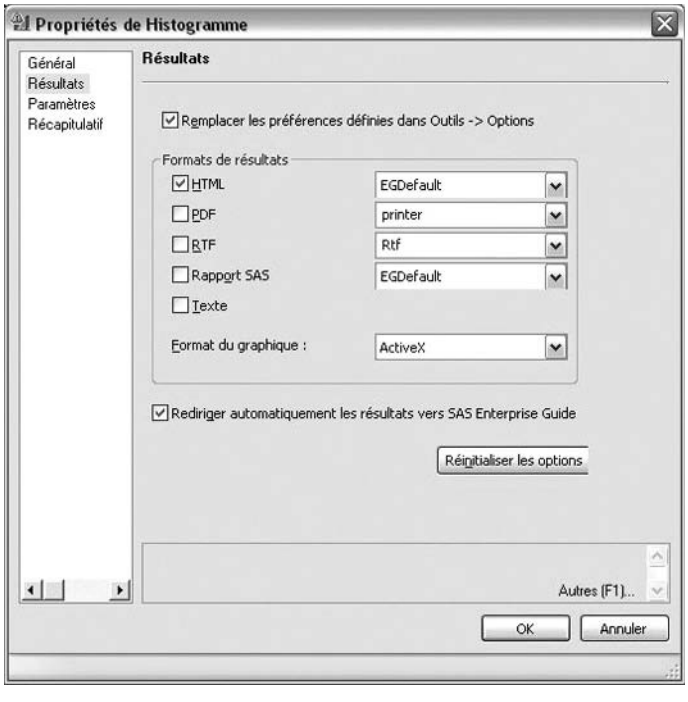


Figure 7.8 — Le menu d'options associé à une tâche dans SEG 4.1

7.4.4 Y a-t-il des précautions à prendre ?

Savoir quel est le serveur (local, serveur Windows, serveur Unix ou serveur MVS) pour connaître les drivers graphiques utilisables. On ne pourra pas utiliser de driver ACTIVEX ni ACTXIMG avec un serveur SAS qui ne serait pas sous Windows. On ne pourra pas activer ODS GRAPHICS ni produire de graphiques GTL (cf. chapitre 8) avec un serveur SAS 9.1 qui ne serait pas sous Windows.

En SEG versions 2 et 3, si on demande la création d'un fichier avec une destination ODS autre que HTML, RTF, PDF ou LISTING, il n'apparaît pas dans le projet au niveau des résultats. En version 4, les fichiers produits par une destination TAGSETS ou MARKUP sont visibles dans le projet, mais doivent être sauvegardés en local (clic droit puis EXPORTER et choisir le premier élément de la liste d'actions proposées) pour être ouverts correctement.

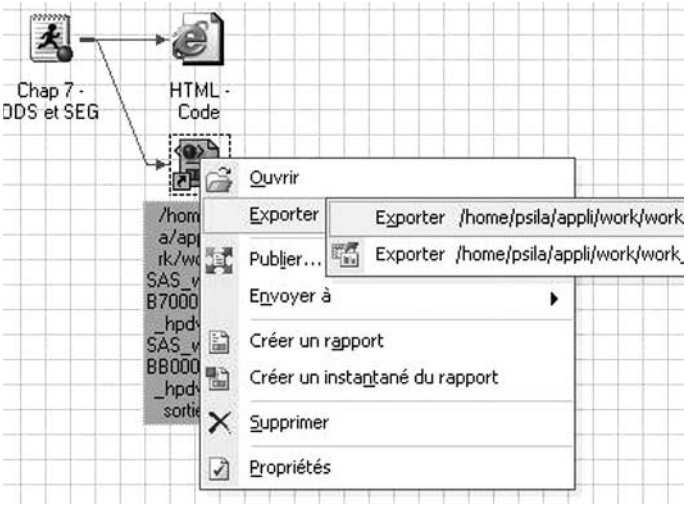


Figure 7.9 — Sauvegarder en local un document XLS créé avec SEG 4.1

7.5 LES APPLICATIONS STOCKÉES (*STORED PROCESS*)

Les applications stockées (en anglais, *stored process*) sont apparues avec la version 8 de SAS, mais réellement mises en avant avec la version 9 version « plateforme BI ». Elles sont un moyen d'appeler depuis n'importe quel logiciel client (Excel, Word, SAS Stored Process Web Application, SAS Enterprise Guide) un programme SAS et de récupérer son résultat dans le même environnement.

7.5.1 Qu'est-ce qu'une application stockée ?

Une application stockée (ou *stored process*) est avant toute chose un programme SAS. Ce programme SAS n'a aucune restriction et peut contenir des procédures SAS, du SQL, du SCL, du langage macro ou encore des étapes Data.

Des instructions ayant trait à l'ODS assureront comme pour un programme SAS classique la restitution. Elles seront le plus souvent *implicites* pour permettre une diffusion plus large et plus hétérogène que dans le contexte d'un programme SAS exécuté sur un poste de travail isolé.

Si une application stockée a beaucoup de points communs avec les programmes SAS, elle s'en différencie par deux points essentiels : le référencement obligatoire dans les métadonnées et l'exécution systématique sur un serveur.

Son intérêt principal tient dans sa capacité d'être exécutée à travers de très nombreuses applications. Ainsi, les applications stockées permettent par exemple à un utilisateur de Microsoft Excel de lancer un programme SAS complexe et d'en récupérer après exécution le résultat dans sa feuille Excel.

Cet avantage est également source de complexité dans la gestion de l'ODS : comme le logiciel dans lequel le résultat peut être rendu n'est pas unique (ça peut être Word, Excel, une page HTML… tout cela à partir de la même application stockée !), il faut que les instructions ODS s'adaptent. Elles le font automatiquement grâce au macro-programme `%STPBEGIN` qui débute la définition de l'application stockée.

Nous supposerons dans la suite de ce chapitre que vous savez créer une application stockée et l'enregistrer auprès des métadonnées.

7.5.2 Où créer des fichiers avec l'ODS ?

Dans une application stockée, on ne précisera pas d'emplacement physique pour la sortie ODS. C'est le programme lui-même qui se chargera d'écrire dans un flux (une tuyauterie informatique) débouchant dans l'application qui a fait appel à l'application stockée. Cette partie est totalement prise en charge par le macro-programme `%STPBEGIN` et ne doit donc faire l'objet d'aucune instruction dans le code.

7.5.3 Peut-on agir sur les options de l'ODS ?

Bien qu'une part importante des commandes ODS soit prise en charge automatiquement, il peut être utile d'ajouter des options supplémentaires. Pour cela, on peut utiliser des instructions `ODS` sans option `FILE`. Mais puisque, selon le client appelant l'application stockée, la destination ODS peut changer, comment savoir quelle instruction ODS utiliser ?

On dispose d'une macro-variable `_ODSDEST` qui contient la destination actuellement utilisée par l'application stockée lors de son exécution. On peut s'en servir dans les instructions ODS « rectificatives » (avec la syntaxe `ODS &_ODSDEST` *`options`* `;`).

Cette macro-variable peut également servir à décider d'inclure ou non des options de style dans une procédure Print, Report ou Tabulate (par exemple, ne pas utiliser l'option de style `FLYOVER` si la destination est RTF).

La macro-variable `_ODSDEST` peut prendre des valeurs dépendant du client qui demande l'exécution. Par exemple :

- à partir de SAS Enterprise Guide, `HTML`, `RTF` ou `PDF` ;
- à partir de Add-In For Microsoft Office (Word, PowerPoint ou Excel), `TAGSETS.SASREPORT11` (un XML particulier) ;
- à partir de SAS Stored Process Web Application, `HTML`.

Exemple 7.04 – Gérer le code SAS exécuté en fonction de la destination ODS

```
PROC REPORT DATA=commun.voitures ;
  COLUMNS constructeur modele ;
  DEFINE constructeur / DISPLAY ;
  DEFINE modele / DISPLAY ;
```

```
  COMPUTE modele ;
     IF "&_odsdest"="HTML" THEN
          CALL DEFINE(_col_, "style",
            "style=[PREHTML='<MARQUEE DIRECTION=RIGHT WIDTH=''50%''>'
                   POSTHTML='</MARQUEE>'") ;
  ENDCOMP ;
RUN ;
```

Dans l'exemple 7.04, on ajoute des balises `MARQUEE` (texte défilant) si la sortie est produite au format HTML seulement.

D'autres informations sont également disponibles à travers des macro-variables gérées par l'application stockée. Elles sont résumées dans le tableau 7.1. Il existe également un grand nombre de macros variables réservées qui ne concernent pas toutes l'ODS ou les options de graphiques. À titre d'exemple, certaines permettent de gérer les packages relatifs au SAS Publishing Framework, d'autres sont spécifiques à l'utilisation de SAS Stored Process Web Application. Nous ne les mentionnerons pas plus en détail ici.

Tableau 7.1 — Macro-variables gérées automatiquement dans une application stockée

Macro-variable	Valeur (exemple)	Signification
_ODSDEST	RTF	Destination ODS activée lors de l'exécution de l'application stockée
_ODSSTYLE	DEFAULT	Style ODS utilisé
_ODSSTYLESHEET	(URL="file:///C:\style.css")	Emplacement d'une feuille de style externe (fichiers HTML uniquement)
_ODSENCODING	LATIN1	Codage binaire du jeu de caractères
_ODSOPTIONS	STARTPAGE=NO	Options supplémentaires associées à l'instruction ODS ouvrante
_GOPTION_DEVICE	GIF	Valeur de l'option graphique DEVICE (*driver* graphique ; cf. section 4.1)
_GOPT_HSIZE	10cm	Valeur de l'option graphique HSIZE (taille horizontale du graphique)
_GOPT_VSIZE	10cm	Valeur de l'option graphique VSIZE (taille verticale du graphique)
_GOPT_XPIXELS	400	Valeur de l'option graphique XPIXELS (résolution horizontale du graphique)
_GOPT_YPIXELS	400	Valeur de l'option graphique YPIXELS (résolution verticale du graphique)
_GOPTIONS	FTEXT="Arial"	Autres options graphiques à positionner

Si on souhaite modifier la valeur de ces macro-variables (avec un `%LET`), il faut le faire *avant* l'instruction `%STPBEGIN`. Attention cependant en modifiant ces valeurs : l'application stockée risque d'y perdre son caractère universel. Par exemple, s**i on modifie la valeur de** `_ODSDEST` **(**`%LET _ODSDEST=RTF ;`**)** pour demander à l'application stockée de générer une sortie au format RTF quel que soit le client, l'application stockée n'est, dès lors, plus exécutable depuis Excel.

Exemple 7.05 – Modifier les macro-variables réservées d'une application stockée

```
%LET _goptions = FTEXT="Swiss" ;
%LET _gopt_device = SASEMF ;
%STPBEGIN ;
  PROC GCHART DATA = livre.voitures ;
    PIE type ;
  RUN ; QUIT ;
%STPEND ;
```

7.5.4 Y a-t-il des précautions à prendre ?

Dans le programme de l'application stockée, il ne faut pas laisser d'espace entre le commentaire ProcessBody et le point-virgule. On écrira `*ProcessBody;` mais surtout pas `*ProcessBody ;` !

Si on modifie la macro-variable `_ODSDEST`, il est également préférable de redéfinir les macro-variables `_ODSOPTIONS` et de `_ODSENCODING` à blanc ; on s'assure ainsi que les valeurs de ces macro-variables réservées restent compatibles avec une sortie dans la destination demandée explicitement.

On fera attention, comme dans toute application stockée, à n'utiliser des données que situées dans des bibliothèques accessibles (en termes de droits) au *user* chargé d'exécuter les applications stockées.

Il faut impérativement sélectionner comme serveur d'exécution un serveur d'applications stockées (LOGICAL STORED PROCESS SERVER) si votre application stockée doit produire une sortie ODS. Cette manipulation se fait lors de l'enregistrement de l'application stockée dans les métadonnées (étape 4 de l'assistant).

8

Les grandes avancées disponibles dans SAS 9.2

Objectifs

Ce chapitre traite de deux nouveautés apparues de manière *expérimentale* en version 9.1 : les graphiques écrits en langage GTL et les rapports produits depuis une étape Data avec l'objet ODSOUT[1]. « Expérimental » indiquant chez SAS du code dont la syntaxe n'est pas figée, il nous semblait plus logique de nous contenter d'annoncer ces nouveautés, en présentant des exemples de résultats afin que le lecteur juge si ces points sont susceptibles de l'intéresser, plutôt que d'en détailler par le menu une syntaxe risquant de devenir obsolète au fil des versions futures.

Enfin, ce chapitre présente brièvement les capacités de la procédure GKPI, apparue directement avec la version 9.2 : cette procédure permet de produire des graphiques habituellement disponibles dans les tableaux de bord, portails et autres *balance scorecards* – feux tricolores, compteurs de vitesse, jauges, etc.

8.1 LES GRAPHIQUES EN GTL

La version 9.1.3 donne accès à un langage expérimental de mise en forme de graphiques, le GTL ou *Graph Template Language*. C'est une alternative aux procédures

1. Le site de l'auteur (http://www.od-datamining.com) sera régulièrement mis à jour de documents concernant la manière de coder ces deux types de sorties extrêmement souples et esthétiques.

SAS/GRAPH vues au chapitre 4. Dans la version 9.2, le GTL passe au statut « production » et ne devrait plus voir sa syntaxe changer. Il est assorti de quatre nouvelles procédures qui lui sont dédiées : SGPLOT, SGSCATTER, SGPANEL et SGRENDER.

8.1.1 Le GTL : pourquoi et comment

Le besoin du GTL est apparu concomitamment à l'apparition d'`ODS GRAPHICS` (cf. section 4.9) : pour que les procédures statistiques de SAS produisent des graphiques, il faut leur en donner des définitions d'une manière ou d'une autre. Plutôt que d'inclure ces définitions dans le code sous-jacent à la procédure et de limiter les retouches de l'utilisateur à quelques options, les développeurs de SAS ont préféré les écrire dans un langage neuf et transparent, permettant à tous de se construire des graphiques statistiques sur mesure.

Produits par un driver Java spécifique, les graphiques écrits en GTL ont une esthétique particulière. Ils sont également insensibles aux `GOPTIONS` usuelles, et doivent avoir leurs caractéristiques (leur taille, par exemple) définies dans l'instruction `ODS GRAPHICS` (nettement enrichie en options dans SAS 9.2).

Le fonctionnement du GTL est lui aussi assez spécifique : on définit dans un premier temps un modèle de graphique éventuellement paramétré avec la procédure TEMPLATE (figure 8.1). Puis on utilise ce modèle à travers une étape DATA, une procédure statistique, ou une des quatre procédures spécifiques : SGPLOT, SGSCATTER, SGPANEL et SGRENDER. Dans SAS 9.1.3, seules l'étape Data et les procédures statistiques permettent de produire des graphiques définis en GTL.

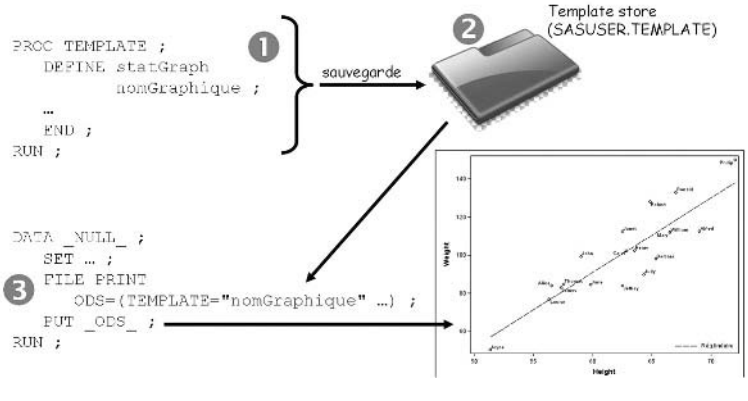


Figure 8.1 — Stockage et utilisation d'un graphique défini en GTL

La syntaxe du GTL étant actuellement en cours d'évolution, les exemples qui sont présentés dans la section suivante ont été écrits pour la version 9.1.3. Il n'est pas garanti qu'ils fonctionneront tels quels en 9.2 ; une documentation en ligne (sur

http://support.sas.com et sur les sites de l'auteur et de l'éditeur) complète et éclaire les exemples de résultats ci-dessous.

8.1.2 Des exemples de graphiques GTL avec SAS 9.1.3

Comme au chapitre 5, la procédure TEMPLATE permet de définir un modèle, ici de graphique. On donne aux modèles graphiques des noms, généralement à 2 niveaux ; cela n'a rien d'obligatoire, mais permet de s'organiser plus aisément, le premier niveau étant un répertoire de l'*items store*.

```
PROC TEMPLATE ;
   DEFINE STATGRAPH nomModele < / STORE = bibliothèque.itemStore > ;
   < DYNAMIC|MVAR paramètre1 < paramètre2 < … > > ; >
     LAYOUT OVERLAY | GRIDDED | LATTICE < / option(s) > ;
         …
     ENDLAYOUT ;
   END ;
RUN ; QUIT ;
```

Les instructions `DYNAMIC` et `MVAR` permettent d'indiquer la présence de paramètres associés au modèle. `MVAR` donne le nom de macro-variables dont la valeur sera récupérée à l'exécution du modèle, au cours de l'étape Data qui y fera appel. `DYNAMIC` indique des noms de paramètres (sorte de variables texte internes au modèle) dont l'étape Data devra proposer des valeurs. L'instruction `LAYOUT` décrit selon quel *motif* s'organisent plusieurs éléments graphiques au sein du même modèle – s'ils sont superposés ou juxtaposés.

Exemple 8.01 – Boxplots en GTL

```
PROC TEMPLATE ;
  DEFINE STATGRAPH exemples.boxplot ;
    DYNAMIC varY varGpe id ;
    LAYOUT GRIDDED ;
      BOXPLOT Y=varY X=varGpe / DATALABEL=id ;
    ENDLAYOUT ;
  END ;
RUN ;
PROC SORT DATA = livre.voitures OUT = work.voitures ;
       BY americaine ;
RUN ;
ODS HTML GPATH="c:\temp" ;
DATA _NULL_ ;
  SET work.voitures ;
  nom = CATX(" ",constructeur, modele) ;
  rpp = poids * 1000 / puissance ;
  LABEL rpp = "Rapport poids / puissance du véhicule (kg/cv)" ;
  FILE PRINT ODS=(TEMPLATE="exemples.boxplot"
                  DYNAMIC=(varY="rpp" varGpe="americaine" id="nom")) ;
  PUT _ODS_ ;
RUN ;
ODS HTML CLOSE ;
```

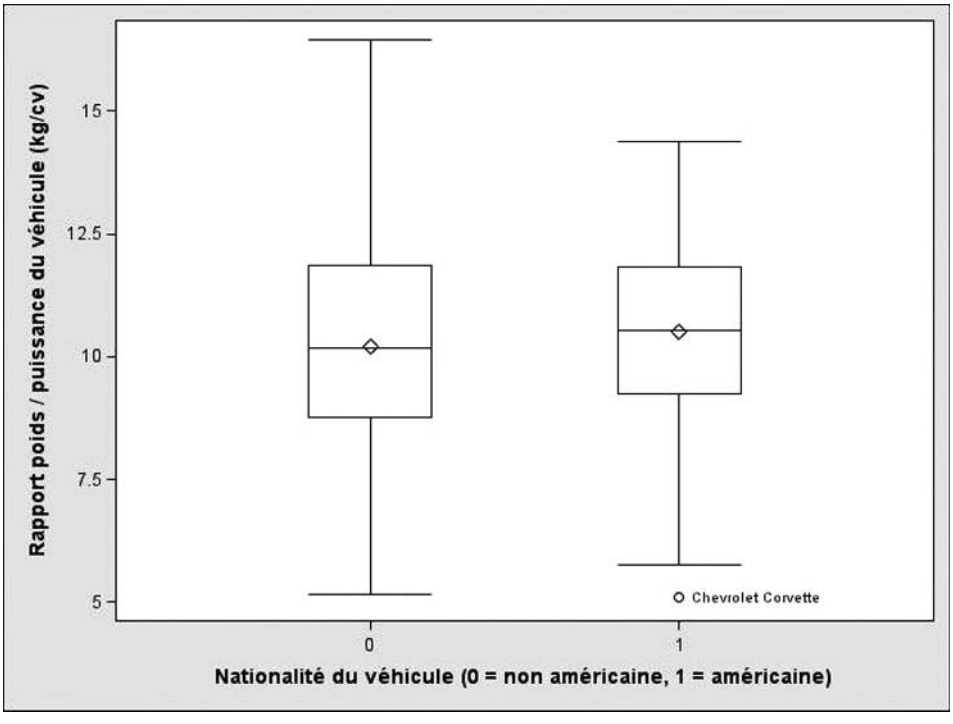


Figure 8.2 — Résultat de l'exemple 8.01 (device Java implicite)

Exemple 8.02 – ACP en GTL (reprise de l'exemple 4.18)

```
PROC TEMPLATE ;
  DEFINE STATGRAPH mesGraphs.acp ;
    LAYOUT OVERLAY / XGRID=TRUE YGRID=TRUE
                     XAXISOPTS=(LABEL="Axe n°1" TICKS=(-1 0 1))
                     YAXISOPTS=(LABEL="Axe n°2" TICKS=(-1 0 1)) ;
      VECTORPLOT X=prin1 Y=prin2 / DATALABEL=variable
                                   XMIN=-1 XMAX=1 YMIN=-1 YMAX=1 ;
      ELLIPSEPARM SEMIMAJOR=1 SEMIMINOR=1 SLOPE=0 / LINEPATTERN=DASH ;
    ENDLAYOUT ;
  END ;
RUN ;
ODS OUTPUT eigenVectors = work.vp ;
PROC PRINCOMP DATA = livre.voitures ;
  VAR puissance poids conso_auto conso_ville longueur ;
RUN ;
DATA _NULL_ ;
  SET work.vp ;
  FILE PRINT ODS=(TEMPLATE="mesGraphs.acp") ;
  PUT _ODS_ ;
RUN ;
```

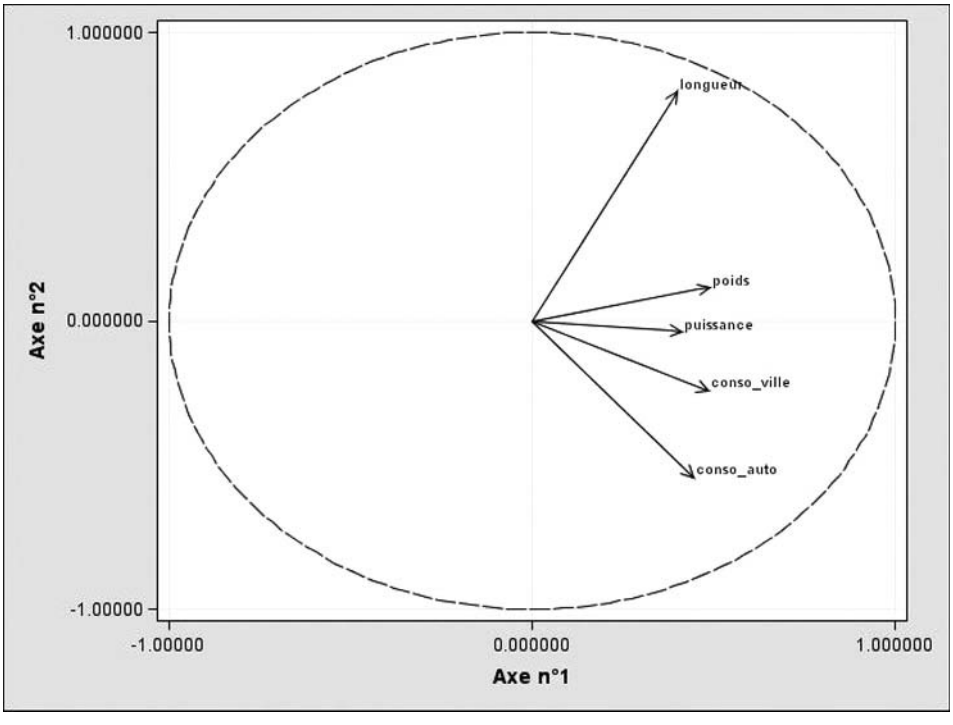


Figure 8.3 — Résultat de l'exemple 8.02 (device Java implicite)

Exemple 8.03 – Quatre histogrammes côte à côte en GTL

```
PROC TEMPLATE ;
  DEFINE statGraph exemples._4histogrammes ;
    DYNAMIC varY1 varY2 varY3 varY4 ;
    LAYOUT GRIDDED / ROWS=2 COLUMNS=2 ;
      HISTOGRAM varY1 / XAXISOPTS=(LABEL=varY1) ;
      HISTOGRAM varY2 / XAXISOPTS=(LABEL=varY2) ;
      HISTOGRAM varY3 / XAXISOPTS=(LABEL=varY3) ;
      HISTOGRAM varY4 / XAXISOPTS=(LABEL=varY4) ;
    ENDLAYOUT ;
  END ;
RUN ;
DATA _NULL_ ;
  SET work.voitures ;
  FILE PRINT ODS=(TEMPLATE="exemples._4histogrammes"
                  DYNAMIC=(varY1="citadine" varY2="compacte"
                           varY3="familiale" varY4="sportive")) ;
  PUT _ODS_ ;
RUN ;
```

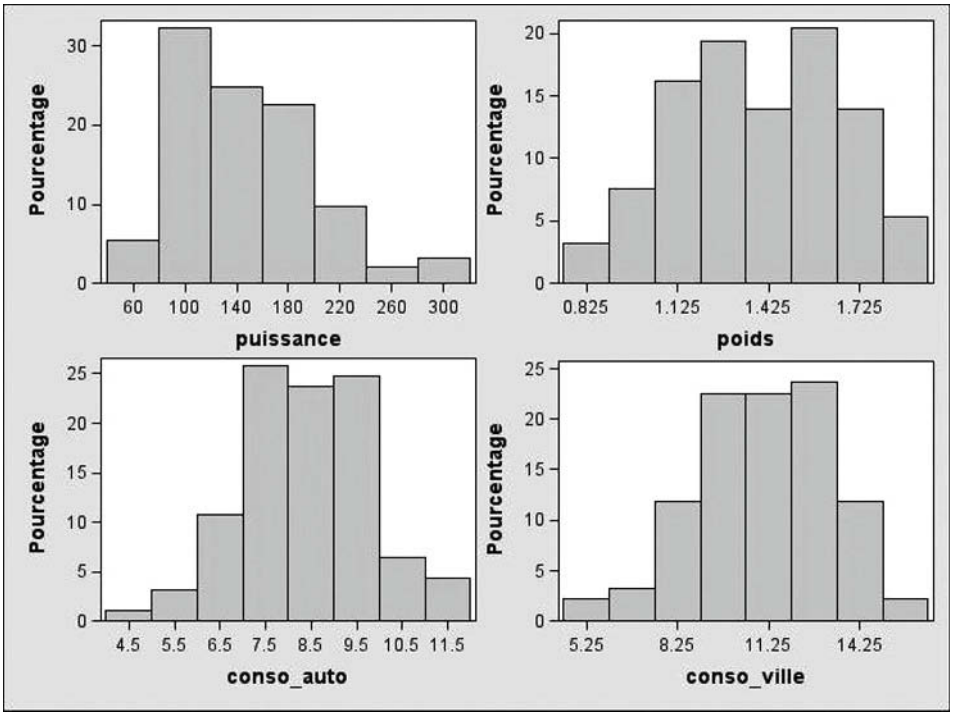


Figure 8.4 — Résultat de l'exemple 8.03 (device Java implicite)

8.1.3 Des exemples de graphiques GTL avec SAS 9.2

La version 9.2 apporte quelques changements substantiels à la syntaxe des exemples précédents. Par exemple, pour que les exemples 8.01, 8.02 et 8.03 donnent des résultats corrects avec SAS 9.2, il faudra ajouter une ligne `BEGINGRAPH ;` avant l'instruction `LAYOUT` et une ligne `ENDGRAPH ;` après `ENDLAYOUT`.

Mais les avancées les plus fondamentales autour du GTL tiennent à la présence de quatre nouvelles procédures, SGPLOT, SGSCATTER, SGRENDER et SGPANEL. SGPLOT et SGSCATTER sont des hybrides des procédures GPLOT, GCHART, UNIVARIATE et BOXPLOT avec une esthétique améliorée. Elles offrent en outre la possibilité de superposer aisément plusieurs éléments sur un graphique, sans table Annotate, et d'ajouter des libellés aux points d'un nuage avec un dispositif anti-collision (figure 8.5). SGRENDER concurrence l'étape Data pour afficher un graphique défini par la proc TEMPLATE comme dans les programmes précédents.

SGPANEL est particulièrement impressionnante car elle permet de produire plusieurs graphiques côte à côte, en nombre variable en fonction des modalités d'une ou de plusieurs variables.

Exemple 8.04 – Nuage de points avec la procédure SGPLOT (version 9.2)

```
DATA work.reperes ;
  DO poids = 750 TO 2000 BY 250 ;
    sportive = poids / 6 ; nerveuse = poids / 10 ; OUTPUT ;
  END ;
RUN ;
DATA work.voitures ;
  SET livre.voitures (IN=donnees) work.reperes ;
  IF donnees THEN poids = poids * 1000 ;
  LABEL poids = "Poids (kg)" ;
RUN ;
ODS GRAPHICS ON / IMAGEFMT=GIF ;
ODS HTML GPATH="c:\temp" ;
PROC SGPLOT DATA=work.voitures NOAUTOLEGEND ;
  BAND X=poids LOWER=sportive UPPER=300 / FILLATTRS=(COLOR="WHITE")
                                LEGENDLABEL="Sportive" NAME="bande1" ;
    BAND X=poids LOWER=nerveuse UPPER=sportive / FILLATTRS=(COLOR="GRAY90")
                                LEGENDLABEL="Nerveuse" NAME="bande2" ;
  BAND X=poids LOWER=0 UPPER=nerveuse  / FILLATTRS=(COLOR="GRAYC0")
                                LEGENDLABEL="Normale" NAME="bande3" ;
  SCATTER Y=puissance X=poids / DATALABEL=modele ;
  KEYLEGEND "bande1" "bande2" "bande3" ;
RUN ;
ODS HTML CLOSE ;
```

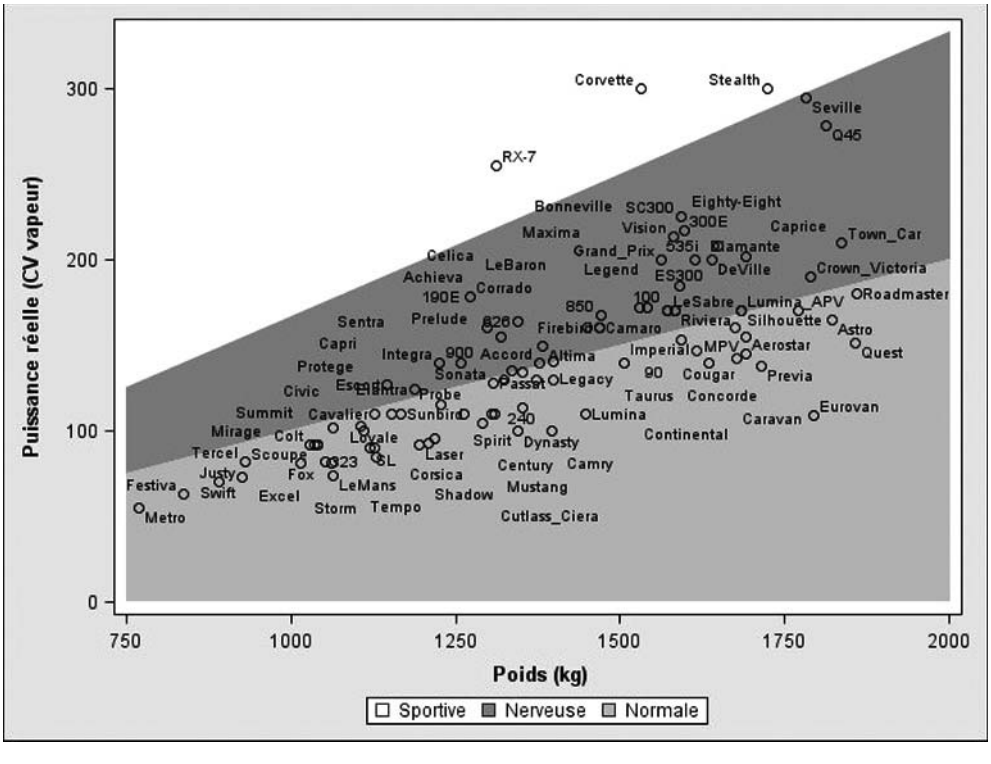


Figure 8.5 — Résultat de l'exemple 8.04 (driver Java, image au format GIF)

Exemple 8.05 – Histogrammes côte à côte avec la procédure SGPANEL (version 9.2)

```
ODS GRAPHICS ON / IMAGEFMT=GIF ;
ODS HTML GPATH="c:\temp" ;
PROC SGPANEL DATA=livre.voitures ;
  PANELBY americaine transmission / LAYOUT=LATTICE;
  LABEL americaine = "Américaine ?" ;
  HISTOGRAM conso_ville ;
RUN ;
ODS HTML CLOSE ;
```

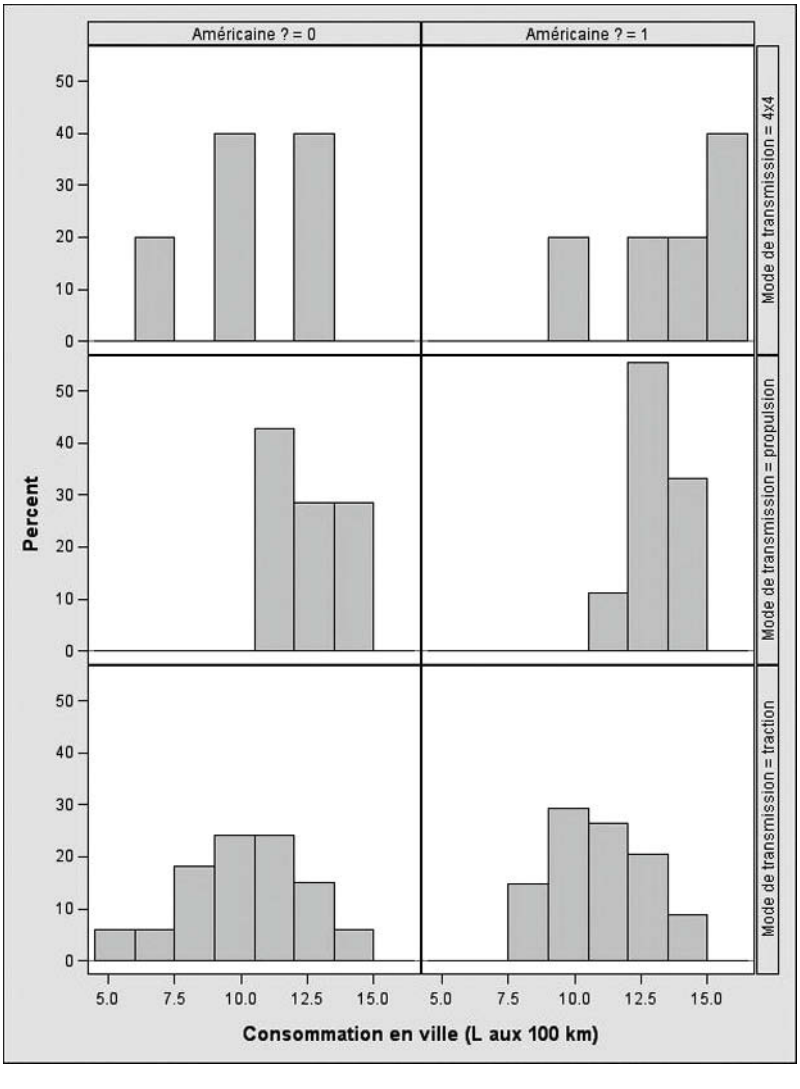


Figure 8.6 — Résultat de l'exemple 8.05 (driver Java, image au format GIF)

8.2 LES TABLEAUX SUR MESURE AVEC L'OBJET ODSOUT

La création de rapports sur mesure est longtemps passée uniquement par l'étape DATA : les instructions `FILE` et `PUT` existaient bien avant les procédures TABULATE et REPORT, pour l'export de fichiers plats mais aussi (`FILE PRINT`) pour l'édition de texte dans la fenêtre Output. Cependant, ce mode de fonctionnement limité à une police à pas fixe (type « SAS Monospace ») tend à se raréfier, tandis que le besoin de rapports complexes s'intensifie. Avec la version 9.1.3 apparaît l'objet `ODSOUT` dans l'étape Data, qui permet de construire des rapports constitués de tableaux dans la destination ODS ouverte (toutes sauf LISTING, DOCUMENT et OUTPUT).

L'objet ODSOUT n'est pas compatible avec la destination LISTING. Il faut donc fermer celle-ci quand on exécute une étape Data contenant l'objet ODSOUT en version 9.1, sous peine de générer des messages d'erreur.

La programmation orientée objet est assez rare dans SAS, aussi voici quelques petites choses à savoir sur un tel mode de programmation…

- Un objet est une structure de données informatique destinée à un rôle particulier. Il peut accueillir un grand nombre d'informations simultanément.
- Un objet doit être initialisé (ou instancié) ; l'instruction utilisée dans SAS se nomme `DECLARE`.
- Un objet voit ses propriétés modifiées par des fonctions qu'on lui applique. Ces fonctions portent le nom de *méthodes*. On les applique à l'objet avec une syntaxe « pointée » : `nomObjet.nomMéthode(arguments de la méthode)`. Le résultat d'une méthode est l'objet lui-même.

```
ODS LISTING CLOSE ; /* indispensable sous peine d'erreurs */
ODS HTML|RTF|PDF FILE="…" options ;
DATA _NULL_ ;
   SET tableSAS ;
   IF _N_=1 THEN DECLARE ODSOUT nomObjet () ;
   …
RUN ;
ODS HTML|RTF|PDF CLOSE ;
ODS LISTING ;
```

8.2.1 Une structure tabulaire écrite ligne à ligne…

À l'intérieur de l'étape DATA, on initialise donc l'objet `ODSOUT`. Les méthodes que l'on peut principalement lui appliquer sont ensuite : `TABLE_START` et `TABLE_END` qui délimitent un tableau dans le document produit ; `ROW_START` et `ROW_END` qui délimitent chaque ligne du tableau produit ; `FORMAT_CELL` qui indique le contenu de chaque cellule (argument `TEXT`). On y indique également les mises en forme (éléments de

style) particulières que l'on souhaite avoir sur cette cellule, avec l'argument OVERRIDES.

La logique de l'objet ODSOUT est donc de permettre toutes les fantaisies à chaque ligne d'un tableau, puisque la construction se fait ligne par ligne. La logique ici est analogue à celle qui décrit les tableaux dans le langage HTML.

Contrairement aux procédures PRINT, REPORT et TABULATE, une colonne du tableau final peut contenir des données sans cohérence : dates à certaines lignes, texte à d'autres, etc.

```
DATA _NULL_ ;
   SET tableSAS END=fin ;
   IF _N_=1 THEN DECLARE ODSOUT nomObjet () ;
   nomObjet.TABLE_START() ; /* début du tableau */
    nomObjet.ROW_START() ; /* début d'une ligne */
      nomObjet.FORMAT_CELL(TEXT: "contenu de cellule") ;
      nomObjet.FORMAT_CELL(TEXT: "contenu de cellule") ;
      nomObjet.FORMAT_CELL(TEXT: "contenu de cellule") ;
    nomObjet.ROW_END() ; /* fin de la ligne */
    …
   nomObjet.TABLE_END() ;
RUN ;
```

Dans cette syntaxe, il ne faut pas perdre de vue que l'étape DATA et l'instruction SET construisent automatiquement une boucle sur les données, et que les méthodes seront appliquées à chaque observation lue. On utilisera donc des IF, ainsi que la variable _N_ et celle créée par l'option END (booléen vrai quand on atteint la dernière observation de la table) pour déclencher des méthodes au début et à la fin des données. Comme on le verra avec l'exemple 8.07, on peut également utiliser les marqueurs de blocs, FIRST. et LAST., que crée l'instruction SET…BY sur une table SAS triée.

Enfin, il est indispensable de se rappeler que chaque contenu de cellule est vu comme du texte pour l'objet ODSOUT. Il est donc indispensable de transformer les variables numériques à l'aide de la fonction PUT (ou de sa variante PUTN) lors de l'application de la méthode FORMAT_CELL.

Exemple 8.06 – Rapport sur mesure avec ODSOUT (version courte)

```
ODS _ALL_ CLOSE ;
ODS PDF FILE = "c:\temp\data _null_.pdf" ;
DATA _NULL_ ;
  SET livre.voitures END=fin ;
  IF _N_=1 THEN DO ;
    DECLARE ODSOUT listing () ; /* initialisation */
    listing.TABLE_START() ;
  END ;
  listing.ROW_START() ;
    listing.FORMAT_CELL(TEXT:CATX(" ", constructeur,modele),
                        OVERRIDES:"FONT_WEIGHT=BOLD
                                   BACKGROUND=BLACK FOREGROUND=WHITE") ;
    listing.FORMAT_CELL(TEXT:CATX(" ", "Ventes :",ventes),
```

```
                        OVERRIDES:"BACKGROUND=BLACK FOREGROUND=WHITE") ;
  listing.ROW_END() ;
  listing.ROW_START() ;
    listing.FORMAT_CELL(TEXT:"Type") ;
    listing.FORMAT_CELL(TEXT:type) ;
  listing.ROW_END() ;
  listing.ROW_START() ;
    listing.FORMAT_CELL(TEXT:"Transmission") ;
    listing.FORMAT_CELL(TEXT:transmission) ;
  listing.ROW_END() ;
  IF fin THEN listing.TABLE_END() ;
RUN ;
ODS PDF CLOSE ;
ODS LISTING ;
```

Acura Integra	Ventes : 576
Type	Citadine
Transmission	traction
Acura Legend	**Ventes : 263**
Type	Familiale
Transmission	traction
Audi 90	**Ventes : 409**

Figure 8.7 — Résultat de l'exemple 8.06, fichier DATA _NULL_.PDF

8.2.2 … où l'on peut fusionner des cellules et inclure des images

Ce mode de fonctionnement où l'on construit ligne à ligne un tableau permet également de fusionner des cellules sur plusieurs colonnes ou plusieurs lignes. C'est l'objet d'arguments `ROW_SPAN` et `COLUMN_SPAN` dans la méthode `FORMAT_CELL`. Il est également possible d'insérer des images avec la méthode `IMAGE`, des sauts de page avec la méthode `PAGE`, des lignes de séparation avec `LINE`.

Exemple 8.07 – Rapport sur mesure avec ODSOUT (version longue)

```
ODS LISTING CLOSE ;
PROC SORT DATA = livre.voitures OUT = work.voitures ;
  BY constructeur modele ;
RUN ;
ODS PDF FILE = "c:\temp\data _null_.pdf" ;
DATA _NULL_ ;
  SET work.voitures (DROP = ventes) END=fin ;
  BY constructeur ;
  ARRAY caracteristiques _NUMERIC_ ;
  IF _N_=1 THEN DECLARE ODSOUT listing () ; /* initialisation */
  IF FIRST.constructeur THEN DO ;
    listing.TABLE_START(OVERRIDES:"FRAME=VOID RULES=NONE") ;
      listing.ROW_START() ;
        listing.FORMAT_CELL(TEXT:
                  CATX(" ","Extrait de la gamme",constructeur,"..."),
                  OVERRIDES:"BACKGROUND=BLACK FOREGROUND=WHITE") ;
      listing.ROW_END() ;
    listing.TABLE_END() ;
    listing.IMAGE(TEXT:CATS("c:\temp\img\",LOWCASE(constructeur),".gif")) ;
  END ;
  listing.TABLE_START() ;
    listing.ROW_START() ;
      listing.FORMAT_CELL(TEXT:
                          CAT("Modèle : ",STRIP(constructeur)," ",modele),
                          COLUMN_SPAN:4,
                          OVERRIDES:"JUST=C BACKGROUND=GRAY
                                     FONT_WEIGHT=BOLD BOTTOMMARGIN=2mm") ;
    listing.ROW_END() ;
    DO i=1 TO DIM(caracteristiques) ;
      IF MOD(i,2)=1 THEN listing.ROW_START() ;
      listing.FORMAT_CELL(TEXT:VLABEL(caracteristiques(i))) ;
      listing.FORMAT_CELL(TEXT:
                  PUTN(caracteristiques(i), VFORMAT(caracteristiques(i)))) ;
      IF MOD(i,2)=0 THEN listing.ROW_END() ;
    END ;
  listing.TABLE_END() ;
  IF NOT LAST.constructeur THEN listing.LINE() ;
  IF LAST.constructeur AND NOT fin THEN listing.PAGE() ;
RUN ;
ODS PDF CLOSE ;
ODS LISTING ;
```

Extrait de la gamme Acura ...

Modèle : Acura Integra			
Prix d'entrée de gamme (FF)	77400	Prix max - Prix min / 2 (FF)	95400
Prix du haut de gamme (FF)	112800	Consommation en ville (L aux 100 km)	9.41
Consommation sur autoroute (L aux 100 km)	7.59	Nombre d'airbags	0
Nombre de cylindres	4	Cylindrée (L)	1.8
Puissance réelle (CV vapeur)	140	Régime de puissance maxi (tr/min)	6300
Existance d'un modèle à boîte manuelle	1	Volume du réservoir (L)	49
Nombre de passagers	5	Longueur du véhicule (cm)	449
Empattement du véhicule (cm)	259	Largeur du véhicule (cm)	172
Rayon de braquage (m)	11	Espace aux places arrières (cm)	67
Volume du coffre (cm3)	180	Poids du véhicule (tonnes)	1.225

Modèle : Acura Legend			
Prix d'entrée de gamme (FF)	175200	Prix max - Prix min / 2 (FF)	203400
Prix du haut de gamme (FF)	232200	Consommation en ville (L aux 100 km)	13.07

Figure 8.8 — Résultat de l'exemple 8.07, fichier DATA _NULL_.PDF

8.3 LES INDICATEURS DE PERFORMANCE

La procédure GKPI (*Graphic Key Performance Index*, soit principaux indicateurs graphiques de performance) apparaît avec SAS version 9.2. Contrairement aux deux éléments de langage précédents (GTL et objet `ODSOUT`), il s'agit d'une procédure qui apparaît directement en production. Sa syntaxe n'est donc pas sujette à changements futurs. Cette procédure produit des graphiques que l'on trouve traditionnellement dans les applications de tableaux de bord : jauges, compteurs, feux tricolores. Ici, on ne lit pas de table SAS (il n'y a pas d'option `DATA`) mais on indique la valeur à faire porter sur l'instrument d'affichage (feu, jauge, tachymètre) dans une option `ACTUAL`. L'option `MODE=RAISED` permet d'ajouter au graphique des effets de relief. Cette procédure n'est compatible qu'avec le driver JAVAIMG.

Exemple 8.08 – Jauge et tachymètre avec la procédure GKPI

```
PROC SQL NOPRINT ;
  SELECT MEAN(conso_ville < 10),    MEAN(conso_ville)
    INTO : pct_eco_ville,           : moy_ville
  FROM livre.voitures
  ;
QUIT ;
```

```
GOPTION DEVICE=JAVAIMG ;
PROC GKPI MODE=RAISED ;
  SPEEDOMETER ACTUAL=&pct_eco_ville
              BOUNDS=(0 .40 .60 1) / TARGET=.75
              LFONT=(F="Arial") LABEL="% de voitures économiques en ville"
              FORMAT="PERCENT8.0" ;
RUN ; QUIT ;
PROC GKPI MODE=RAISED ;
  VSLIDER ACTUAL=&moy_ville
          BOUNDS=(0 5 10 15 20) /
                 COLORS=(GREEN ORANGE YELLOW RED)
          LABEL="Conso moyenne en ville (L/100 km)"
          FORMAT="NUMX12.2" ;
RUN ; QUIT ;
GOPTIONS RESET=ALL ;
```

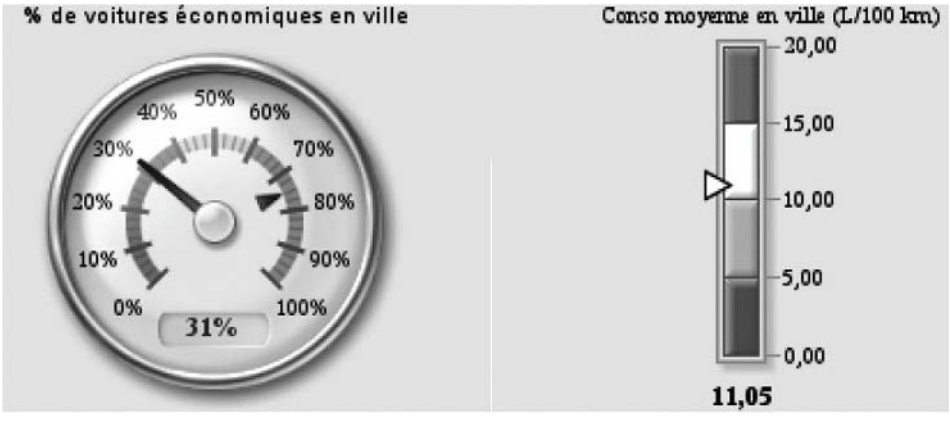


Figure 8.9 — Résultats de l'exemple 8.08 (device Javalmg)

Références

Ce livre s'est construit autour de mon expérience personnelle, des questions posées au cours de formations ou de missions de conseil, de pratiques vues ici et là, et de documents glanés sur Internet. Voici quelques pistes si vous souhaitez plus de précisions sur tel ou tel sujet.

Un livre

- *Carpenter's Complete Guide to the Report Procedure*, Art Carpenter, SAS Press Series. Il est en anglais, il est complet, il parle de la procédure Report, de ses pièges, de ses avantages, de son rapport avec l'ODS. Un CD de programmes est inclus dans le livre, très pédagogique dans son approche et son propos.

Des sites Internet

- **http://support.sas.com/forums/index.jspa –** Sur le site US de SAS Institute, des forums d'échange entre utilisateurs, avec la participation du personnel de SAS à l'occasion (dont l'épatante Cynthia Zender, formatrice chez SAS US). Une catégorie est réservée aux questions de *reporting* (*ODS and Base Reporting*). Il faut être inscrit pour poster une question ou une réponse (l'inscription est gratuite et prend 30 s), mais la consultation est libre.
- **http://www.lexjansen.com/sugi/ –** La monstrueuse collection de communications au SUGI (*SAS Users Group International*, s'appelle désormais SAS Global Forum) – la réunion annuelle des utilisateurs. Les nombreux papiers traitant de l'ODS et du reporting en général sont aussi bien produits par des développeurs (pratique pour comprendre le fonctionnement ou l'idée derrière une syntaxe) que par des utilisateurs (on y glane en général des astuces et des savoir-faire inégalables). Un moteur de recherche se trouve sur la page d'accueil. C'est un moyen simple de trouver moult documents sur une thématique en utilisant des mots-clés (par ex. « ODS RTF » renvoie 254 réponses).

Des documents

Dans la liste suivante, les documents PDF qui n'ont pas d'adresse Internet associée peuvent être trouvés sur le site de Lex Jansen (http://www.lexjansen.com/sugi/) *via* le moteur de recherche de la page d'accueil.

- **Sur l'ODS en général**
  - Un résumé de l'ODS en version 9.1, http://support.sas.com/rnd/base/ods/templateFAQ/ODS91.pdf
  - *Creating Complex Reports*, Cynthia L. Zender.
  - *Reporting avec SAS 9.1 : de la proc Print à ODS Escapechar*, Olivier Decourt, disponible sur http://www.od-datamining.com/ (rubrique *télécharger*, avec un fichier ZIP des programmes).
- **Sur ODS TAGSETS.EXCELXP**
  - *Creating and importing multi-sheet Excel Workbooks the Easy Way with SAS*, Vincent DelGobbo (présente ODS TAGSETS.EXCELXP).
  - *L'export de SAS vers Excel expliqué à ma fille*, Olivier Decourt, disponible sur http://www.od-datamining.com/ (rubrique *télécharger*).
- **Sur l'inline formatting**
  - *Smooth Writing with In-Line Formatting*, Louise S. Hadden.
  - *Enhancing RTF Output with RTF Control Words and In-Line Formatting*, Lori S. Parsons.
  - *Creating a Customized Table of Contents in ODS RTF Documents*, Electra Small.
  - *Rich Text Format (RTF) Version 1_5 Specification*, document de référence Microsoft, disponible sur http://www.biblioscape.com/rtf15_spec.htm ou sur le site de Microsoft.
- **Sur les procédures Report et Tabulate**
  - *So you want to learn Proc Report*, par Chris Moriak
  - *Practically Perfect Presentations Using ODS and PROC REPORT*, Cynthia L. Zender.
  - *PROC REPORT : Doin'It in Style !*, Ray Pass.
  - *So You're Still Not Using PROC REPORT. Why Not ?*, Ray Pass et Daphne Ewing.
  - *Creating Complex Reports*, Cynthia L. Zender.
  - *Using PROC REPORT to Generate "Impossible" Totals*, Andrea J. Decker.
  - *Beyond the Basics: Advanced PROC REPORT Tips and Tricks*, Allison McMahill.
  - *La proc TABULATE, sa vie, son œuvre*, Olivier Decourt, disponible sur http://www.od-datamining.com/ (rubrique *télécharger*, dans la série « Expliqué à ma fille »).

- **Sur les procédures graphiques de SAS/GRAPH**
  - *Exporting SAS/GRAPH® Output for Inclusion in Web Pages and Other Software Applications*, Warren Repole Jr.
  - *An Introduction to Exporting SAS/Graph Output to Microsoft Office SAS Release 8.2 and higher*, document de travail SAS Institute n°TS-674, disponible sur http://support.sas.com/techsup/technote/ts674/ts674.html
  - *Enhancements to SAS/GRAPH® Software in SAS 9.2*, Himesh Patel.
  - *La proc GPLOT expliquée à ma fille*, Olivier Decourt, disponible sur http://www.od-datamining.com/ (rubrique *télécharger*).
  - *La proc GCHART expliquée à ma fille*, Olivier Decourt, disponible sur http://www.od-datamining.com/ (rubrique *télécharger*).
  - *La proc GREPLAY expliquée à ma fille*, Olivier Decourt, disponible sur http://www.od-datamining.com/ (rubrique *télécharger*).

- **Sur la procédure Template**
  - *PROC TEMPLATE Tables from Scratch*, Kevin D. Smith.
  - *SAS with Style: Creating your own ODS Style Template for RTF Output*, Lauren Haworth.
  - *Pure Evil, or Just Misunderstood ? An Interview with PROC TEMPLATE*, Kevin P. Delaney.
  - *Customize your SAS® Output with the Template Procedure : A Beginning Tutorial*, Carol Gosselin et Joy Munk Smith.
  - *The TEMPLATE Procedure Styles : Evolution and Revolution*, Kevin D. Smith.
  - *Creating RTF-Output Using ODS and PROC TEMPLATE*, Marcel Hoevenaars et Machteld Baljet.

- **Sur SAS BI**
  - *Adapting Your Programs to the SAS® 9 Paradigm*, Cynthia L. Zender.
  - *ODS Options and SAS® Stored Processes*, Cynthia L. Zender.

- **Sur les graphiques GTL**
  - *A Programmer's Introduction to the Graphics Template Language*, Jeff Cartier.
  - *Standard Graph Templates*, Philip R Holland.
  - *The Power of the Graphics Template Language*, Jeff Cartier.
  - *SAS/GRAPH® Procedures for Creating Statistical Graphics in Data Analysis*, Dan Heath.
  - *Butterflies, Heat Maps, and More : Using the SAS/GRAPH® Procedures SGPLOT and SGPANEL*, Susan Schwartz.

- **Sur l'objet ODSOUT**
  - *Next Generation Data _NULL_ Report Writing Using ODS OO Features*, Daniel O'Connor.
  - *Experimenting with the ODS DATA Step Object*, Richard Koopmann Jr. (deux documents).
  - *Using New Features in ODS to Create Master/Detail Reports*, Jack Hamilton

Index

D

E

F

O

P

Q

R

S

T

U

V

W

X

Y

INFOPRO

TYPE D'OUVRAGE

L'ESSENTIEL	SE FORMER	RETOURS D'EXPÉRIENCE

Olivier Decourt
Préface de Philippe Letren

- MANAGEMENT DES SYSTÈMES D'INFORMATION
- **APPLICATIONS MÉTIERS**
- ÉTUDES, DÉVELOPPEMENT, INTÉGRATION
- EXPLOITATION ET ADMINISTRATION
- RÉSEAUX & TÉLÉCOMS

REPORTING AVEC SAS

Mettre en forme et diffuser vos résultats avec SAS 9 et SAS 9 BI

Cet ouvrage s'adresse aux étudiants et aux professionnels ainsi qu'à tous ceux qui ont à mettre en forme et à diffuser des tableaux, des rapports et des graphiques créés avec SAS.

Le logiciel SAS se place sans conteste parmi les leaders des marchés de **l'informatique décisionnelle**, de la *Business Intelligence* et de l'analyse statistique. Mais il a également la réputation de produire des sorties assez peu esthétiques, et de demander beaucoup d'efforts pour arriver à éditer un graphique diffusable.

Les explications données dans ces pages sur les différentes procédures de **mise en forme avancée des sorties** font évoluer cette opinion sur SAS : logiciel puissant, il permet non seulement **l'analyse des données** mais aussi **la diffusion de résultats** clairs et compréhensibles. Cet ouvrage contient plus d'une centaine d'exemples qui vont faciliter le quotidien du programmeur. Les données et les programmes de ces exemples sont disponibles sur Internet.

Les versions les plus récentes (SAS v8, SAS 9 et SAS 9 BI) fourmillent de fonctionnalités pour produire aisément de très belles sorties, avec les outils les plus communs de l'entreprise – traitement de texte, tableur, pages web. Celles qui sont le plus susceptibles de trouver une **utilité quotidienne** dans des programmes ont été sélectionnées et sont minutieusement détaillées dans ce livre. Enfin, pour mettre en lumière les avancées les plus prometteuses, un chapitre présente les principales nouveautés de la **version 9.2**.

OLIVIER DECOURT

Est formateur et consultant indépendant sur SAS, SPAD, l'analyse statistique et le Data Mining. Il enseigne SAS dans plusieurs universités (IUT STID, ENSAI…).

SUPPLÉMENTS en ligne

Télécharger les sources de tous les exemples de l'ouvrage ainsi que des compléments d'information sur www.dunod.com et sur www.od-datamining.com.

ISBN 978-2-10-053575-0

www.dunod.com

DUNOD