LANGAGE SAS

Axelle Chauvet-Peyrard

Année scolaire 2006-2007

Présentation

SAS (Statistical Analysis System) est un logiciel de traitement de l'information conçu en 1976 par une firme américaine, The SAS Insitute. Il est organisé en de multiples modules ; outre le module de base, on abordera ici principalement les modules SAS/STAT (procédures de traitement statistique) et SAS/GRAPH (procédures graphiques).

Cet ouvrage répond à deux objectifs :

- Initier les débutants au langage SAS, en en expliquant la logique et en présentant la syntaxe des principales instructions et fonctions de base.
- Fournir un outil de travail concret, en référençant de nombreuses procédures. Pour chaque procédure, on décrit ce qu'elle fait ainsi que ses principales options et instructions ; des exemples d'utilisation sont ajoutés, ainsi qu'une aide à l'interprétation des sorties générées.

L'ouvrage est particulièrement adapté à la version 8 de SAS. Certains des outils ou des procédures abordés peuvent ne pas être appliqués à des versions antérieures du logiciel, ou bien à des versions partielles (dans lesquelles il manquerait les modules associés à ces outils). Cependant, chaque fois que cela est nécessaire, on proposera des alternatives aux outils présentés.

INTRODUCTION : Présentation générale du logiciel SAS et de ses objets

Il s'agit d'un rappel de définition des objets SAS (bibliothèques, tables, catalogues) et de l'interface (les différentes fenêtres de travail, les raccourcis clavier…).

PARTIE I : L'étape DATA

L'étape DATA sert principalement à la manipulation des tables de données. C'est l'élément de base de la programmation en SAS. Cette partie permet d'apprendre à écrire des étapes DATA. Après une explication détaillée de la logique de programmation, on rappelle les syntaxes des principales options et instructions de manipulation des données.

PARTIE II : L'étape PROC

L'étape PROC est une mise en œuvre de procédures pré-programmées dans le logiciel. Cette partie commence par présenter le schéma général d'une étape PROC, avant de détailler une vingtaine de procédures parmi les plus utilisées, et réparties en trois grands groupes :

A : Procédures de manipulation des données
B : Procédures de traitement statistique
C : Procédures graphiques

PARTIE III : Aide à l'utilisation du logiciel

Cette partie est axée sur l'utilisation pratique du logiciel. Elle se découpe en trois sous-parties :

A : Conseils pratiques (aide en ligne, interprétation de la LOG, gestion des erreurs)
B : Des assistants pour gagner du temps (présentation d'outils presse-bouton sous SAS v8, alternatives à la programmation de certaines procédures)
C : Sauvegarder et exporter des sorties SAS (les solutions pour enregistrer et récupérer en dehors du logiciel les listings et graphiques générés par les procédures)

PARTIE IV : Introduction aux macro-programmes SAS

Les macro-programmes sont des blocs d'instructions et de procédures SAS dont la réalisation dépend d'un ou plusieurs paramètres passés en argument du macro-programme. Cet outil, qui rappelle les procédures de la programmation classique, permet de gagner beaucoup de temps au quotidien. L'objectif de cette partie est d'apprendre à utiliser des macros SAS et à en écrire soi-même de simples.

Annexes

Ces annexes doivent permettre une recherche aisée dans les pages du polycopié. Une double approche est proposée :

- Un index alphabétique des concepts statistiques et mots-clés SAS abordés dans l'ouvrage.
- Un index thématique sur le modèle de : « je veux faire ça » → « voici comment le faire ».

Sommaire

INTRODUCTION

Présentation générale du logiciel SAS et de ses objets

Tables, catalogues et bibliothèques :

SAS travaille sur des données regroupées en tableaux qu'il appelle des tables (Data) : en colonne figurent les variables, en en ligne les observations ou individus.

➢ *Définition et allocation d'une bibliothèque (Library) :*

Une bibliothèque est un objet SAS qui consiste en un nom virtuel que l'on associe à un répertoire physique.
La première chose à faire pour pouvoir accéder aux tables stockées dans un répertoire (par exemple \\c:\sas\) est d' « allouer une bibliothèque » sur ce répertoire. Deux méthodes :

- Par l'icône New Library (Nouvelle Bibliothèque)
- Ou l'instruction suivante : `libname malib "c:\sas";`

Cette instruction crée une bibliothèque de nom malib qui pointe sur le répertoire physique \\c:\sas\.
La bibliothèque permet d'accéder à tous les objets SAS contenus dans ce répertoire : tables de données, mais aussi catalogues !

➢ *Définition d'un catalogue (Catalog) :*

Un catalogue est un objet SAS qui peut contenir des formats ou des macros compilés, des graphiques sous format image, des listings, etc. qui deviennent ainsi aisément transportables. Cet objet est très utile pour communiquer certains résultats ou outils sans avoir à divulguer le code source du programme). Voir à ce sujet la partie III C.

➢ *Options de l'instruction LIBNAME :*

Une bibliothèque est associée à une version de SAS. Sous SAS v8, vous créez des bibliothèques v8, qui ne peuvent lire que des objets v8. Si vous disposez de tables créées en format v6, vous devez allouer une bibliothèque en forçant le moteur (champ « engine ») V6. Avec l'instruction LIBNAME, il suffit d'écrire :

```
libname v6 malib "c:\sas";
```

Pour désallouer une bibliothèque, on peut utiliser l'option clear :

```
libname malib clear;
```

➢ *Noms des tables de données et bibliothèque par défaut :*

Le nom d'une table doit être précédé du nom de la bibliothèque dans laquelle elle se trouve. Ainsi, malib.matable désigne la table qui porte le nom matable dans la bibliothèque malib. Si aucun nom de bibliothèque n'est précisé, il est fait implicitement appel à la bibliothèque par défaut de SAS, qui s'appelle WORK. Cette bibliothèque est associée à un dossier temporaire du disque dur, et réinitialisée à la fin de chaque session SAS (ce qui signifie que les données qui y sont stockées sont perdues lorsqu'on quitte l'application).

➢ *Taille limite d'une table*

Il n'y a pas de limite au nombre de lignes. Le nombre maximal de colonnes est de 32 767. A noter que cette limite n'existe plus dans la version 9.

Les différentes fenêtres :

ENHANCED EDITOR (EDITEUR) : Fenêtre dans laquelle vous écrivez votre programme.

LOG (JOURNAL) : Cette fenêtre sert à communiquer les messages d'erreur, avertissements, compte-rendus de la soumission de votre programme.

OUTPUT (SORTIE) : Fenêtre qui contient les sorties issues des étapes PROC.

EXPLORER (EXPLORATEUR) : Contient l'arborescence de vos librarys et de vos objets SAS (tables, catalogues…). Vous pouvez visualiser les propriétés d'un objet en faisant un Clic droit – Properties dessus.

RESULTS (RESULTATS) : Liste des sorties OUTPUT générées. Vous pouvez accéder directement à une sortie présente dans la fenêtre OUTPUT en double cliquant sur un item de cette liste.

Extensions des fichiers obtenus avec SAS :

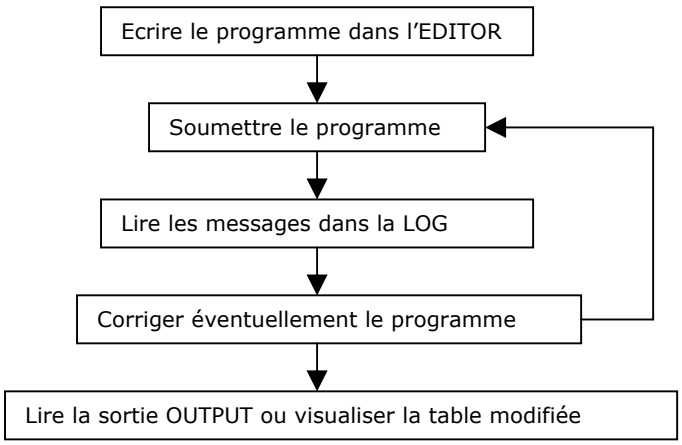
Programme SAS	.sas
Table SAS v8	.sas7bdat
Catalogue v8	.sas7bcat
Sortie Output	.lst
Sortie Log	.log
Table SAS v6	.sd2

Si la bibliothèque malib (format v8) pointe sur le répertoire \\c:\sas\, la table malib.matable correspondra au fichier physique \\c:\sas\matable.sas7bdat. De même, le catalogue malib.moncatlg correspondra au fichier physique \\c:\sas\moncatlg.sas7bcat.

Déroulement de votre travail :

Il existe deux types de programme SAS :

- **Les étapes DATA : pour créer et manipuler les données**
- **Les étapes PROC : essentiellement pour exploiter les données**



Menus et raccourcis clavier :

Soumettre un programme	Run – Submit	F3
Rappeler le dernier programme soumis	Run – Recall Last Submit	F4
Accéder à la fenêtre EDITOR	View – Enhanced Editor	F5
Accéder à la fenêtre LOG	View – Log	F6
Accéder à la fenêtre OUTPUT	View – Output	F7
Accéder à la fenêtre EXPLORER	View – Contents only	
Accéder à la fenêtre RESULTS	View – Results	
Accéder à la fenêtre GRAPH	View – Graph	

Pour passer d'une fenêtre à l'autre, vous pouvez tout simplement cliquer sur l'onglet correspondant à la fenêtre qui vous intéresse, ou bien utiliser Ctrl+Tab.

Vider le contenu d'une fenêtre	Edit - Clear all	Ctrl+E

Vous pouvez aussi taper « clear » dans la barre de commande. Attention : cette commande vide la fenêtre active !

PARTIE I

L'étape DATA

Une étape DATA est un programme de syntaxe :

```
DATA malib.matable ;
  [instructions] ;
RUN ;
```

Un programme de ce type crée une table qui a pour nom matable et appartient à la bibliothèque malib. On rappelle que si aucune bibliothèque n'est spécifiée, la table est placée dans la bibliothèque temporaire Work.

Le fonctionnement d'un tel programme s'effectue par le biais du **vecteur de travail**. L'étape DATA travaille **ligne par ligne**.

Explicitons le principe de l'étape DATA à l'aide de l'instruction de recopie de tables : SET.

Soit le programme suivant :

```
DATA a ;
  SET b ;
RUN;
```

DATA a : Une table a est créée.
SET b : La première ligne de la table b est placée dans le vecteur de travail.
RUN : Marque la fin du programme et son exécution. C'est à ce moment (et pas avant) que le contenu du vecteur de travail est enregistré dans la table a, ligne 1.
Puis, tant qu'il reste des observations dans la table b, l'opération est répétée.

On n'insiste jamais assez sur l'importance des points virgule. Ils indiquent à SAS la fin d'une instruction.
Admettons que dans l'exemple ci-dessus on omette le premier ;
Alors SAS comprend qu'il doit créer trois tables nommées a, set et b !

La logique « ligne par ligne » de l'étape DATA

Soit le programme suivant :

```
DATA maLib.a ;
  total=0;
  SET maLib.cours ;
  Coefficient=coefficient+1 ;
  Un=1 ;
  Total=total+coefficient;
RUN ;
```

Une table a est créée dans la librairie maLib.

Première étape : dans le vecteur de travail une variable total apparaît qui vaut 0

total
0

Puis est recopiée la première ligne de b.

table malib.cours

coefficient	v1	v2
1	4	6
2	3	5
3	5	4

vecteur de travail

total	coefficient	v1	v2
0	1	4	6

table malib.a

Puis au champ coefficient est ajouté 1. Ensuite on y ajoute une variable nommée un qui vaut 1.

table malib.cours

coefficient	v1	v2
1	4	6
2	3	5
3	5	4

vecteur de travail

total	coefficient	v1	v2	un
0	2	4	6	1

table malib.a

Enfin, on affecte à total sa valeur actuelle augmentée de la valeur de coefficient.

table malib.cours

coefficient	v1	v2
1	4	6
2	3	5
3	5	4

vecteur de travail

total	coefficient	v1	v2	un
2	2	4	6	1

table malib.a

On a pour le moment travaillé uniquement sur le vecteur de travail, qui contient la seule ligne courante.

RUN enregistre le vecteur de travail en 1e ligne de la table a et réinitialise le vecteur de travail.

vecteur de travail

table malib.cours

coefficient	v1	v2
1	4	6
2	3	5
3	5	4

table malib.a

total	coefficient	v1	v2	un
2	2	4	6	1

Si bien qu'au début de l'étape 2, total vaut 0, puis on récupère la ligne 2 de b, on incrémente le coef, on ajoute la colonne un, on met à jour total avec le coef courant.

Voilà donc le résultat du programme :

table malib.a

total	coefficient	v1	v2	un
2	2	4	6	1
3	3	3	5	1
4	4	5	4	1

En conclusion on voit qu'il est impossible de calculer de cette manière le cumulé de coefficient, puisque d'une ligne sur l'autre, SAS oublie les valeurs précédentes. On verra plus loin qu'il existe un moyen de calculer des cumulés, grâce à l'instruction RETAIN (voir paragraphe « Calcul de variables »).

Tableau récapitulatif des opérateurs

Opérateurs de comparaison

LT	<	Inférieur strict (lesser than)
GT	>	Supérieur strict (greater than)
LE	<=	Inférieur (lesser or equal)
GE	>=	Supérieur (greater or equal)
EQ	=	Egal (equal)
NE	^=	Différent (not equal)
IN	Signifie l'appartenance à la liste de valeurs indiquée après	

Opérateurs logiques

AND	&	et
OR	!	ou
NOT	^	non

Opérateurs arithmétiques

Les opérateurs naturels +, -, * et /
Ainsi que ** l'opérateur d'exponentiation
a<>b ou Min(a,b) désigne le minimum entre a et b
a><b ou Max(a,b) désigne le maximum entre a et b
Mentionnons enfin l'instruction de sommation : var + 2 équivaut à var = var + 2

Opérateur de concaténation

!!

Recopie d'une table
Sélection de variables et d'observations

La recopie d'une table : Instruction SET

Comme nous l'avons vu précédemment, le programme :

```
DATA destination;
  SET source;
RUN;
```

Crée une table destination dans laquelle on recopie le contenu de la table source.

La sélection de variables : Instructions KEEP et DROP

Il est possible de ne recopier que certaines variables de la table source, en utilisant les instructions et options KEEP et DROP.

Supposons que la table source possède 3 variables nommées v1, v2 et v3. Les programmes suivants :

```
DATA destination;
  SET source;
  KEEP v1;
RUN;
```

```
DATA destination;
  SET source;
  DROP v2 v3;
RUN;
```

```
DATA destination;
  SET source (KEEP=v1);
RUN;
```

```
DATA destination;
  SET source (DROP=v2 v3);
RUN;
```

Ont tous pour effet de créer une table destination qui contient toutes les observations de la table source mais en ne gardant que la colonne v1.

Les deux premiers programmes utilisent les instructions KEEP et DROP, tandis que les deux suivants utilisent ces commandes en tant qu'options de l'instruction SET. Placé en option, le KEEP ou le DROP est exécuté au moment même de la recopie de la table, alors qu'en instruction, il est exécuté après. Si la différence ne se fait pas sentir dans le cas d'une simple recopie de table, comme ici, on verra plus loin un exemple ou elle devient capitale.

Le filtrage d'observations : Instructions WHERE, DELETE et OUTPUT

Il est possible de ne recopier que certaines observations de la table source. Deux cas se présentent :

- *Cas 1 : On souhaite extraire les observations m à n*

On utilise dans ce cas les options OBS= et FIRSTOBS= de l'instruction SET.

```
DATA destination;
  SET source (firstobs=m obs=n);
RUN;
```

Crée une table destination qui contient les observations m à n de la table source.
Si l'option FIRSTOBS n'est pas précisée, la table est recopiée à partir de la 1ère observation. De même, si l'option OBS n'est pas précisée, la table est recopiée jusqu'à la dernière observation.

- *Cas 2 : On souhaite garder les observations vérifiant une certaine condition Cond*

On utilise dans ce cas l'instruction WHERE :

```
DATA destination;
  SET source;
  WHERE [Cond];
RUN;
```

Attention : L'instruction WHERE doit être unique dans l'étape DATA. Par exemple, on écrira :

```
DATA destination;
  SET source;
  WHERE v1>0 AND v2="a";
RUN;
```

Et non :

```
DATA destination;
  SET source;
  WHERE v1>0;
  WHERE v2="a";
RUN;
```

En effet, dans ce programme, la seconde instruction WHERE écrase la première !

De manière équivalente, on peut utiliser les instructions DELETE (efface l'observation courante) et OUTPUT (force l'écriture de l'observation courante dans la table) dans une boucle si…alors… :

```
DATA destination;
  SET source;
  IF [Cond] THEN OUTPUT;
RUN;
```

```
DATA destination;
  SET source;
  IF not [Cond] THEN DELETE;
RUN;
```

Là encore, dans un cadre simple comme celui-ci, ces méthodes sont équivalentes. On verra par la suite qu'elles ne le sont pas toujours.

L'instruction OUTPUT est en particulier très utile lorsqu'on souhaite « découper » une table en plusieurs sous-tables. Par exemple, je dispose d'une table source dont une des variables est le sexe de l'individu, et je souhaite obtenir deux tables, l'une contenant les données pour les hommes, et l'autre pour les femmes. J'écris alors le programme suivant :

```
DATA femmes hommes;
  SET source;
  IF sexe=1 THEN OUTPUT hommes;
  ELSE OUTPUT femmes;
RUN;
```

Remarques sur l'instruction SET

- La table destination et la table source peuvent être identiques. Dans ce cas, l'ancienne table source est écrasée par la nouvelle.
- L'instruction SET peut être utilisée avec plusieurs tables (exemple : SET source1 source2 ;). Dans ce cas, les deux tables sont concaténées.
- Si plusieurs instructions SET sont spécifiées, les tables sont fusionnées.

Au sujet de ces deux dernières remarques, on se reportera au paragraphe « Concaténation et fusion de tables ».

Les attributs des variables

Chaque variable possède les attributs suivants :

Attributs obligatoires :
- **Nom**
- **Longueur de stockage (length)**
- **Type (caractère ou numérique)**

Attributs facultatifs :
- **Label** : champ texte plus long que le nom, qui accepte blancs et caractères accentués, et qui permet de fournir une courte description de la variable ; si un label est défini, c'est lui qui est affiché lors de la visualisation de la table.
- **Format de lecture (format)** : la longueur de stockage étant définie, on peut appliquer à la variable un format de lecture, qui correspond à une manière de présenter les valeurs de cette variable ; le format est utile dans le cas de nombres à décimales ou de dates.
- **Format d'écriture (informat)** : c'est l'équivalent du format, mais il sert au moment de l'écriture des données dans la table et non au moment de leur lecture.

Les formats :

➢ *Formats prédéfinis :*

Les formats numériques s'appliquent aux variables de type numérique.
Le format **n.** indique que la variable numérique sera lue / écrite sur n caractères.
Le format **n.d** indique que la variable numérique sera lue / écrite sur n caractères *dont* d décimales.
Le format **Zn.** force l'écriture sur n caractères en complétant à gauche par des 0.

Les formats caractères s'appliquent aux variables de type caractère, et commencent par un « $ ».
Le format **$n.** indique que la variable caractère sera lue / écrite sur n caractères.

➢ *Cas particulier des formats date :*

En SAS, une date est une variable numérique qui représente le nombre de jours écoulés depuis une certaine date de référence interne au système. Afin de rendre les valeurs plus parlantes, on peut appliquer un des formats prédéfinis suivants (il en existe d'autres) :

Format	*Exemple avec le 14 janvier 2005*
DATE9.	14JAN2005
YYMMDD8.	05-01-14
DDMMYY8.	14/01/05

➢ *Formats définis par l'utilisateur :*

L'utilisateur peut définir ses propres formats et informats grâce à la procédure PROC FORMAT (voir le paragraphe consacré dans la partie II A).
La définition de formats s'avère particulièrement utile lorsqu'on souhaite faire du regroupement de modalités sans modifier les valeurs d'origine.

Note : Tous les formats se terminent par un « . ».

Connaître les attributs d'une variable :

Dans la fenêtre de visualisation de la table (que l'on obtient en double cliquant sur une table dans la fenêtre Explorer), il suffit de faire un clic droit sur l'en-tête de la colonne dont on désire connaître les attributs, puis de choisir « Column Attributes ».

Vous pouvez aussi retrouver tous les attributs de toutes les variables d'une table en visualisant les propriétés de la table. Deux méthodes :
- Dans la fenêtre explorer, clic droit sur la table puis « Properties ».
- Procédure PROC CONTENTS (voir le paragraphe consacré dans la partie II A).

Modifier les attributs obligatoires d'une variable :

➢ *Modifier le nom : Instruction RENAME*

Dans une étape DATA, on utilise l'instruction ou l'option RENAME :

```
DATA destination;
  SET source;
  RENAME old=new;
RUN;
```

```
DATA destination;
  SET source (rename=(old=new));
RUN;
```

Ces deux programmes ont pour effet de créer une table destination qui est la copie de la table source, la variable old ayant été renommée en new. Dans ce cas très simple, le choix d'utiliser l'instruction ou l'option est transparent. On verra plus loin que ce n'est pas toujours le cas.

➢ *Modifier la longueur de stockage : Instruction LENGTH*

A l'intérieur d'une étape DATA, on utilise l'instruction LENGTH :

```
LENGTH mavariable <$> n ;
```

Définit pour la variable de nom mavariable une longueur de stockage de n caractères.
Dans le cas d'une variable caractère, on ajoute un $ entre le nom de la variable et le nombre de caractères n.

Cette instruction ne peut pas être appliquée à une variable déjà existante !
Si l'on souhaite modifier la longueur de stockage d'une variable existante, il faut définir une nouvelle variable de la longueur voulue, puis affecter les valeurs de l'ancienne variable à cette nouvelle variable.

Exemple :

Programme	*Commentaire*
`DATA destination;`	Je crée une table destination
`  SET source;`	Dans laquelle je recopie les données de la table source
`  LENGTH newvar $ 10;`	La variable newvar (inconnue dans la table source) est une variable caractère de longueur de stockage égale à 10
`  newvar = var;`	La variable newvar prend les valeurs de var (appartenant à la table source)
`  DROP var;`	La variable var n'appartient pas à la table destination
`  RENAME newvar=var;`	La variable newvar s'appellera var dans la table destination
`RUN;`	

➢ *Modifier le type : Fonctions PUT() et INPUT()*

Une variable est définie dès le départ comme étant d'un certain type : caractère ou numérique. Cet attribut n'est pas destiné à être modifié. Si l'on souhaite tout de même le faire, il faut, comme précédemment, passer par une nouvelle variable.
Supposons comme exemple que la table source contienne une variable varchar, de type caractère, de longueur de stockage 4, que l'on souhaite transformer en une variable numérique. On écrit le programme suivant :

Programme	*Commentaire*
`DATA destination;`	Je crée une table destination
`  SET source;`	Dans laquelle je recopie la table source
`  FORMAT newvarnum 4.;`	La variable newvarnum (inconnue dans source) est une variable numérique de format 4.
`  newvarnum = input (varchar , 4.);`	Les valeurs prises par newvarnum résultent de la conversion en numérique (et au format 4.) des valeurs de la variable varchar
`RUN;`	

De manière symétrique, la fonction `put(varnum,format_caractère)` permet de convertir des données numériques en caractères.

Modifier les attributs facultatifs d'une variable :

Appliquer un label : Instruction LABEL

A l'intérieur d'une étape DATA, on utilise simplement l'instruction LABEL :

```
LABEL mavariable "ceci est mon label" ;
```

Cette instruction applique le label « ceci est mon label » à la variable mavariable. Si la variable avait déjà un label, il est écrasé par le nouveau.

Appliquer un format ou un informat : Instructions FORMAT et INFORMAT

A l'intérieur d'une étape DATA, on utilise l'instruction FORMAT :

```
FORMAT mavariable monformat ;
```

Cette instruction applique le format monformat à la variable mavariable. Monformat doit être reconnu comme un nom de format valide, soit un format prédéfini, soit un format défini par l'utilisateur. En particulier, ne pas oublier le . à la fin.
L'instruction INFORMAT s'utilise selon la même syntaxe.

L'instruction FORMAT s'utilise aussi au sein d'une étape PROC. Dans ce cas, le format de la variable n'est pas modifié au sein de la table, mais affecte uniquement le traitement en cours. Sur ce sujet, on se reportera au paragraphe consacré à la PROC FORMAT, partie II A.

Concaténation et fusion de tables

La concaténation de tables : Instruction SET

Il s'agit de « coller verticalement » les observations de deux tables a et b afin de n'obtenir qu'une seule table. C'est encore l'instruction SET qui va servir dans ce cas.
Prenons comme exemple les deux tables a et b suivantes :

table a

nom	note
toto	12
titi	14
tata	18

table b

nom	note
titi	10
tutu	8

Alors le programme suivant :

```
DATA ab;
  SET a b;
RUN;
```

Fournit le résultat suivant :

table ab

nom	note
toto	12
titi	14
tata	18
titi	10
tutu	8

Les colonnes portant le même nom sont regroupées en une seule colonne. Les individus de b sont recopiés à la suite des individus de a, sans souci de classement. Pour remédier à cela, on peut effectuer un « interclassement ».

L'interclassement de deux tables : Instructions SET et BY

Il s'agit simplement d'ajouter une instruction BY après l'instruction SET.

```
DATA ab;
  SET a b;
  BY nom;
RUN;
```

Ce programme ne fonctionne que si les deux tables ont auparavant été triées selon la variable nom (ce qui n'est pas le cas ici). Pour trier une table, on utilise une procédure PROC SORT (voir le paragraphe consacré dans la partie II A).

Supposant que l'on ait trié les tables, le programme ci-dessus fournit le résultat suivant :

table ab

nom	note
tata	18
titi	14
titi	10
toto	12
tutu	8

A noter qu'il revient au même de réaliser la concaténation simple sur données non triées puis de trier le résultat (la table ab) à l'aide d'une PROC SORT.

Distinction des données selon leur table d'origine

Supposons maintenant que la note de la table a ait un sens différent de la note de la table b. La simple concaténation des deux tables perd cette information. Deux solutions se présentent à nous :

- *Créer deux variables « note » différentes :*

Il suffit de renommer l'une des deux variables note, comme suit :

```
DATA ab;
  SET a b (rename=(note=note2));
RUN;
```

Ce programme fournit le résultat suivant :

table ab

nom	note	note2
toto	12	.
titi	14	.
tata	18	.
titi	.	10
tutu	.	8

On voit bien dans ce contexte la différence qu'il y a à traiter le changement de nom d'une variable comme option de l'instruction de copie ou comme instruction indépendante. En effet, le programme suivant :

```
DATA ab;
  SET a b;
  RENAME note=note2;
RUN;
```

Aurait donné le résultat suivant :

table ab

nom	note2
toto	12
titi	14
tata	18
titi	10
tutu	8

Soit exactement le même résultat qu'avec le tout premier programme, à ceci près que la deuxième colonne ne porte plus le même nom !

- *Ajouter une colonne type de note :*

La méthode précédente présente un inconvénient majeur, celui de générer de nombreuses valeurs manquantes.
La deuxième méthode consiste à ajouter une colonne indiquant la provenance des observations. Pour cela, on utilise l'option IN= de l'instruction SET :

```
DATA ab;
  SET a (in=x) b;
  IF x=1 THEN origine="a";
  ELSE origine="b";
RUN;
```

Ce programme fournit le résultat suivant :

table ab

nom	note	origine
toto	12	a
titi	14	a
tata	18	a
titi	10	b
tutu	8	b

La fusion de tables : Instruction MERGE

Mais dans un cas tel que celui-ci, où l'on dispose de différentes données sur un même ensemble d'individus, on pourrait vouloir obtenir une table dont chaque observation corresponde à un individu. Ici, on souhaiterait avoir la table suivante :

table ab

nom	note_a	note_b
tata	18	.
titi	14	10
toto	12	.
tutu	.	8

Il s'agit en quelque sorte de « coller horizontalement » les deux tables, ce qui en SAS s'appelle une fusion. On utilise alors l'instruction MERGE.
De plus, ici, on « contrôle » la fusion afin de mettre en concordance les données d'un même individu. Ce contrôle s'effectue grâce à une instruction BY.
Le programme permettant d'obtenir le tableau ci-dessus est alors :

```
DATA ab;
  MERGE a (rename=(note=note_a)) b (rename=(note=note_b));
  BY nom;
RUN;
```

Remarque : Comme chaque fois que l'on utilise une instruction BY, il est nécessaire d'opérer sur des tables déjà triées.

Remarque 2 : Si l'on oublie de renommer les variables, la variable « note » de la table b vient écraser la variable « note » de la table a, et l'on obtient :

table ab

nom	note
tata	18
titi	10
toto	12
tutu	8

Remarque 3 : Si l'on ne contrôle pas la fusion, on a une simple juxtaposition des tableaux. Dans le cas où les tables ne sont pas triées (et à condition de renommer les variables), cela donne :

table ab

nom_a	note_a	nom_b	note_b
toto	12	titi	10
titi	14	tutu	8
tata	18		

La mise à jour d'une table à partir des données d'une autre table : Instruction UPDATE

Supposons maintenant que les notes de la table a correspondent aux notes d'un examen donnée et que les notes de la table b soient celles de l'examen de rattrapage correspondant. Les données de b doivent donc venir compléter, voire corriger, les données de a. On utilise alors l'instruction UPDATE.

Le programme suivant :

```
DATA a_b;
  UPDATE a b;
  BY nom;
RUN;
```

Doit bien entendu être appliqué à a et b *après tri de ces tables* selon le nom.
Il fournit le résultat suivant :

table a_b

nom	note
tata	18
titi	10
toto	12
tutu	8

Titi ayant passé le rattrapage, sa note provenant de a est écrasée par sa note provenant de b.
Tutu n'avait pas passé l'examen initial. Il est ajouté à la table !

Remarque : On obtient ici le même résultat qu'en effectuant une fusion contrôlée sur le nom, dans le cas où l'on ne renomme pas les variables « note ». Les deux méthodes ne sont cependant pas équivalentes. Avec UPDATE, seules les valeurs non manquantes écrasent les anciennes valeurs !

Remarques sur l'instruction BY :

On vient de voir plusieurs cas d'utilisation de l'instruction BY. A ce stade, il convient de remarquer qu'il s'agit d'une instruction très courante, que l'on retrouvera aussi dans le cadre d'étapes PROC (voir partie II).
Il est important de comprendre que **BY en soi-même n'effectue aucun tri**. Si la table issue d'un interclassement, d'une fusion contrôlée ou d'une mise à jour est triée, c'est que les tables en entrée l'étaient ! Le travail de BY consiste à « contrôler » les valeurs de la variable désignée afin de les mettre en concordance. Le but est de repérer les données appartenant à un même individu.
Ainsi, une fusion contrôlée ou une mise à jour doit s'effectuer selon une variable qui permet d'identifier l'individu. Dans l'exemple ci-dessus, on a considéré que le nom pouvait servir d'identifiant. Dans un cas concret, il vaudra mieux éviter (risque d'homonymes).

Boucles DO et conditions IF

Le langage SAS offre la possibilité de faire des boucles finies ou des boucles tant que, ainsi que des tests.

Les conditions IF :

```
IF condition THEN instruction ;
```
(Éventuellement suivi de) `ELSE instruction ;`

Le IF *n'a pas besoin* d'être terminé par un END ;

Si on veut insérer plus d'une instruction dans le THEN ou dans le ELSE, il faut les encadrer par un bloc DO ; … END ; de la manière suivante :

```
IF condition THEN DO ; bloc d'instructions END ;
      ELSE DO ; bloc d'instructions END ;
```

Le bloc SELECT :

Si une série de conditions IF revient à différencier le traitement qui suit selon les modalités d'une variable, on lui substituera avantageusement une instruction SELECT dont voici la syntaxe :

```
SELECT (variable) ;
  WHEN (modalité1) instruction1 ;
  WHEN (modalité2) instruction2 ;
  …
  OTHERWISE instruction ;
END ;
```

Le cas OTHERWISE n'est pas indispensable.

Les boucles finies : (exemple pour 10 itérations)

```
DO i=1 TO 10 ;
  Bloc d'instructions (ne pas oublier les points virgules)
END ;
```

Les boucles tant que :

```
DO WHILE (condition) ;
  Bloc d'instructions à exécuter tant que la condition est réalisée
END ;
```

Les fonctions SAS

Fonctions de manipulation de chaînes de caractères :

Length(x)	Retourne la longueur de x
Compress(x,'c')	Compresse x (en enlevant les caractères c)
Repeat(x,n)	Forme une chaîne de caractères qui est n fois la répétition de x
Index(x,y)	Retourne la place du début du mot y dans x
Upcase(x)	Met x en majuscules
Lowcase(x)	Met x en minuscules
Dequote(x)	Enlève les guillemets présents dans x
Quote(x)	Encadre x de guillemets
Substr(x,n,l)	Extraie de x un mot de longueur l à partir du nième caractère
Scan(x,n,'sp')	Extraie de x le nième mot considérant que sp est le séparateur
Tranwd(x,y,z)	Remplace dans x toutes les occurrences du mot y par le mot z

Fonctions de manipulation de dates :

mdy(m,j,a)	Crée une date de jour j, mois m et année a
Date()	Retourne la date courante
Datepart(d)	Extraie la partie date d'une date d
Day(d)	Retourne le jour d'une date d
Month(d)	Retourne le mois d'une date d
Year(d)	Retourne l'année d'une date d
Weekday(d)	Retourne le jour dans la semaine d'une date d

Fonctions mathématiques :

Floor(x)	Partie entière de x
Abs(x)	Valeur absolue de x
Sign(x)	Vaut 1 si x>0, -1 si x<0, 0 sinon
Round(x,a)	Arrondit x à la précision a
Max(x1,…,xn)	Maximum des valeurs de x1,…xn
Min(x1,…,xn)	Minimum des valeurs de x1,…,xn
Mod(x,y)	Reste de la division euclidienne de x par y
Sqrt(x)	Racine carrée de x
Exp(x)	Fonction Exponentielle
Log(x)	Fonction Logarithme néperien
Cos(x)	Fonction Cosinus
Sin(x)	Fonction Sinus
Tan(x)	Fonction Tangente
Arcos(x)	Fonction Cosinus inverse
Arsin(x)	Fonction Sinus inverse
Atan(x)	Fonction Tangente inverse

Fonctions aléatoires :

Les fonctions suivantes génèrent des nombres selon une loi choisie. L'algorithme nécessite la définition d'un paramètre a (par exemple : 0).

Rannor(a)	Loi normale centrée réduite
Ranuni(a)	Loi uniforme sur [0,1]
Ranpoi(a,l)	Loi de Poisson de paramètre l
Ranbin(a,n,p)	Loi binomiale de paramètres n et p
Rantbl(a,p1,…,pn)	Loi discrète de distribution p1,…,pn

Fonctions statistiques :

n(x1,…,xn)	Nombre de valeurs non manquantes parmi les variables x1,…,xn
nmiss(x1,…,xn)	Nombre de valeurs manquantes parmi les variables x1,…,xn
sum(x1,…xn)	Somme des variables x1,…,xn
mean(x1,…xn)	Moyenne des variables x1,…,xn
var(x1,…xn)	Variance empirique des variables x1,…,xn

Probabilités :

Les fonctions suivantes sont les fonctions de répartition en t des lois désignées.

Probnorm(t)	Loi normale centrée réduite
Probchi(t,n)	Loi du Chi2 à n degrés de liberté
Poisson(t,l)	Loi de Poisson de paramètre l
Probf(t,n,p)	Loi de Fisher de paramètres n et p
Probbnml(p,n,t)	Loi binômiale de paramètres n et p
Probhypr(N,k,n,t)	Loi hypergéométrique de paramètres N, k et n

Fonctions particulières :

Input(vchar,fornum)	Transforme une variable caractère vchar en variable numérique au format fornum
Put(vnum,forchar)	Transforme une variable numérique vnum en variable caractère au format forchar
Lagn(x)	Retourne la valeur n fois précédente de x ; pour n=1 on note Lag(x)
Difn(x)	Retourne x - Lagn(x) ; pour n=1 on note Dif(x)

Calculs de variables

L'opérateur d'affectation est le =. Les calculs les plus simples (addition, soustraction, multiplication, division, exponentiation) se font simplement par invocation de l'opérateur approprié.
Il n'est pas nécessaire de déclarer les variables en SAS. Tout objet du programme qui, de par sa nature syntaxique, est interprété comme un nom de variable, est inséré dans le vecteur de travail en tant que nouvelle variable.
On étudie ci-dessous deux cas particuliers de calcul de variables.

Générer des distributions aléatoires

On souhaite créer une table *ex nihilo*, qui contienne 100 réalisations d'une variable aléatoire réelle x suivant une loi normale centrée réduite. On écrit le programme suivant :

```
DATA normale;
  DO i=1 TO 100;
    x = rannor(0);
    OUTPUT;
  END;
RUN;
```

La table normale ainsi créée contient **deux variables** : i (numéro d'itération) et x (variable suivant une loi normale centrée réduite, obtenue).
L'instruction OUTPUT est ici indispensable. **Si on l'omet, seule la dernière observation est enregistrée dans la table**. Ceci tient au fait que SAS n'assimile pas une étape d'une boucle à une observation. Ainsi, dans le programme suivant :

```
DATA test;
  SET source;
  DO i=1 TO 100;
    [Bloc d'instructions];
  END;
RUN;
```

Le bloc d'instructions est effectué **100 fois pour chaque observation** de la table source !

Calculer des cumulés

On a vu dans la présentation de l'étape DATA que cela n'est pas immédiat, la raison en étant que le vecteur de travail est réinitialisé à chaque nouvelle observation calculée.
Pour y remédier, il faut demander à SAS de se souvenir de la valeur précédente prise par la variable. Cela se fait grâce à l'instruction RETAIN :

```
RETAIN mavariable <valeur_initiale> ;
```

La syntaxe de RETAIN prévoit la définition (facultative) d'une valeur initiale pour mavariable, valeur qui lui sera affectée lors de sa première apparition dans le programme. Si l'on souhaite calculer un cumulé, on choisira cette valeur égale à 0.

On reprend le cas étudié dans le paragraphe « La logique ligne par ligne de l'étape DATA » en ajoutant une instruction RETAIN :

```
DATA maLib.a ;
  RETAIN total 0;
  SET maLib.cours ;
  Coefficient=coefficient+1 ;
  Un=1 ;
  Total=total+coefficient;
RUN ;
```

Cette fois, la variable total contient bien le cumulé des coefficients :

table malib.a

total	coefficient	v1	v2	un
2	2	4	6	1
5	3	3	5	1
9	4	5	4	1

Remarque : L'instruction RETAIN ne peut s'appliquer qu'à une nouvelle variable !

➢ *Calcul de cumulés sur sous-groupes*

On reprend cette fois la table ab issue de l'interclassement des tables {nom,note} du paragraphe « Concaténation et fusion de tables ». On souhaite calculer le cumulé des notes pour chaque élève (même si cela n'a pas grand sens).
Comme précédemment, on utilise l'instruction RETAIN.
On a besoin de plus de repérer la première et la dernière occurrence de chaque nom d'élève. Pour cela, on utilise FIRST et LAST :

```
DATA ab;
  SET ab;
  BY nom;
  RETAIN cumule;
  IF first.nom THEN cumule=0;
  cumule=cumule+note;
RUN;
```

On obtient le résultat suivant :

table ab

nom	note	cumule
tata	18	18
titi	14	14
titi	10	24
toto	12	12
tutu	8	8

➢ *Calcul du numéro d'observation*

Cela peut être vu comme un calcul de cumulé, puisque le numéro d'observation est incrémenté de 1 à chaque ligne. On peut donc écrire, en utilisant l'instruction de sommation :

```
DATA destination;
  SET source;
  RETAIN num_obs 0;
  num_obs + 1;
RUN;
```

Cependant, il existe en SAS une variable notée _**n**_ qui désigne le nombre d'itérations réalisées de l'étape DATA en cours. Dans le cas simple d'une recopie de table, cette grandeur correspond exactement au numéro de l'observation courante. On peut donc simplement écrire :

```
DATA destination;
  SET source;
  num_obs = _n_ ;
RUN;
```

Les vecteurs (array) de variables

Un vecteur sous SAS est un moyen de regrouper différents noms de variables afin d'alléger les traitements portant sur ces variables. C'est un outil pratique lorsqu'on doit appliquer la même opération à un grand nombre de variables.

Pour définir un vecteur, on utilise l'instruction ARRAY :

```
ARRAY nom_vecteur <$> variable1 variable2 variable3 variable4 ;
```

Cette instruction définit un vecteur ayant pour nom nom_vecteur et regroupant les variables variable1, variable2, variable3 et variable4.
Dans ce cas, nom_vecteur(1) désigne variable1, nom_vecteur(2) désigne variable2, etc.

Les variables regroupées dans un même vecteur doivent être de même type. Si elles sont de type caractère, il faut intercaler un signe $ après le nom du vecteur.

Dans un cas comme celui-ci, où les variables ont des noms composés d'une partie caractère identique suffixée par un incrément, on peut directement écrire :

```
ARRAY nom_vecteur <$> variable1 - variable4 ;
```

Le nom du vecteur agit comme une sorte d'alias. Il n'apparaît pas dans le vecteur de travail.

- *Dimension d'un vecteur :*

La fonction **dim()** permet de connaître la dimension du vecteur, ie le nombre de variables qu'il regroupe.
Ici, dim(nom_vecteur) vaut 4.

- *Exemple :*

On dispose d'une table source contenant des variables p1, p2, … jusqu'à p20, qui représentent des prix en francs. On veut convertir tous les prix en euros. On écrit le programme suivant :

```
DATA source;
  SET source;
  ARRAY prix p1-p20;
  DO i=1 TO dim(prix);
     prix(i) = prix(i) / 6,55957;
  END;
RUN;
```

Gestion des erreurs et arrêt conditionnel d'une étape DATA

La variable automatique _**error**_ est un booléen qui vaut 1 si le programme a rencontré une erreur, 0 sinon.

Afficher des messages d'erreur : Instruction ERROR

Même en l'absence d'erreur du programme, on peut être amené à considérer certains cas comme des erreurs (par exemple, un âge négatif). On a alors recours à l'instruction ERROR pour traiter ces cas.

```
ERROR « message d'erreur » ;
```

Cette instruction a pour effet de :

- Affecter la valeur 1 à la variable _error_.
- Editer l'observation courante dans la LOG.
- Afficher le message « message d'erreur » dans la LOG.

Exemple :

```
DATA destination;
  SET source;
  IF age < 0 THEN ERROR "l'âge est négatif";
  ELSE [Bloc d'instructions];
RUN;
```

Stopper une étape DATA : Instruction STOP

On peut demander à ce que l'exécution de l'étape DATA s'arrête lorsqu'une certaine condition est réalisée. On utilise pour cela l'instruction STOP.

Exemple : `IF _error_ THEN STOP;` demande l'arrêt de l'étape DATA si une erreur d'exécution se produit.

On peut aussi introduire un test dans une boucle tant que afin d'éviter les boucles infinies, par exemple :

```
DO WHILE (condition);
  IF _n_>1000000 THEN STOP;
  [Bloc d'instructions];
END;
```

Etape DATA sans création de table

Supposons que l'on doive effectuer un calcul ou une recherche à partir d'une table, que l'on souhaite afficher le résultat, mais que l'on n'ait pas besoin de créer une table en sortie. On utilise alors la variable automatique _**null**_ pour indiquer qu'aucune table ne sera créée, et on l'accompagne à l'intérieur de l'étape d'une instruction PUT qui éditera le résultat demandé dans la LOG.

Exemple :

Reprenons les tables a et b présentées dans le paragraphe « Concaténation et fusion de tables ». Je souhaite afficher dans la LOG le nom du (ou des) élève(s) présent(s) dans les deux tables. C'est une information indicative, qui ne me servira plus par la suite ; il n'est donc pas nécessaire que je crée une table contenant cette donnée. J'écris le programme suivant :

```
DATA _null_;
  SET a (in=x) b (in=y);
  IF x AND y THEN PUT nom;
RUN;
```

PARTIE II

L'étape PROC

On rappelle que la programmation élémentaire sous SAS prend deux formes : soit celle d'une étape DATA, dans laquelle on modifie, recopie, manipule des données, soit celle d'une étape PROC.

Dans cette partie, après avoir exposé la structure générale d'une étape PROC, on se propose de présenter une vingtaine de procédures parmi les plus utilisées. Chaque procédure sera présentée selon le schéma suivant :

- Présentation générale de l'objectif de la PROC.
- Eventuellement, quelques rappels théoriques.
- Syntaxe générale (pas toujours exhaustive).
- Les options et instructions majeures.
- Une lecture de sortie standard.
- Une lecture de sortie élaborée.

Structure générale d'une étape PROC

```
PROC XXX <liste_options_de_la_procédure> ;
   INSTRUCTION1 argument_instruction </liste_options_instruction1> ;
   INSTRUCTIONp argument_instruction </liste_options_instructionp> ;
RUN ;
```

Toute étape PROC commence par le mot-clé PROC suivi du nom de la procédure : XXX. Il est possible de mettre à cet endroit ce que l'on appelle des options de la PROC XXX. Cela est facultatif (d'où les < … >) et les éventuelles options sont séparées par un blanc.
Enfin on met un point-virgule.

Puis vient le corps de la procédure, constitué obligatoirement d'une succession d'instructions choisies parmi les instructions de la procédure.
Par exemple, un test de type IF a=0 THEN delete ; n'a rien à faire dans une PROC !! Par contre on pourra utiliser l'instruction WHERE. Chaque instruction commence par un mot-clé et termine par un point-virgule.
Beaucoup d'instructions offrent des options. Il faut alors séparer l'instruction de la liste des options qui s'y rattachent par un /.

Remarque : Si on enchaîne plusieurs étapes PROC, un seul **run ;** à la fin de toutes suffit.

Remarque 2 : Après certaines procédures, on doit mettre un **quit ;** C'est le cas avec des procédures qui font appel à des modules particuliers, telles les PROC GCHART et GPLOT ou la PROC SQL par exemple. Si l'on omet le QUIT, la table, ouverte par le module, n'est pas fermée, et cela peut générer des erreurs.

➢ *Les options incontournables :*

Nous avons déjà évoqué l'option data= . Rares sont les PROC qui ne l'acceptent pas (l'exemple type étant la PROC FORMAT). Notons qu'**en l'absence de cette option, la procédure travaille sur la dernière table créée**.

Une autre option répandue est l'option noprint, qui demande à ce qu'aucune sortie ne soit imprimée dans l'output. Elle évite de surcharger cette dernière lorsque le besoin ne s'en fait pas sentir (par exemple si on a stocké les résultats dans une table ou si on a exporté la sortie avec l'ods).

➢ *Les instructions incontournables :*

L'instruction VAR permet (dans les procédures où elle apparaît) de **préciser sur quelles variables on travaille**.
On la retrouve notamment dans les procédures MEANS, UNIVARIATE, CORR… Elle est donc surtout associée à des traitements de variables quantitatives. Ainsi, elle n'apparaît pas dans la PROC FREQ (qui utilise l'instruction TABLES), ni dans les procédures de modélisation (qui utilisent l'instruction MODEL).

L'instruction BY permet de **constituer des sous-populations**.
Lorsqu'une instruction BY mavar ; est spécifiée dans une PROC, cela force la réalisation d'autant d'étapes PROC qu'il y a de modalités de mavar.
Remarque : Ne pas confondre avec l'instruction CLASS qui permet de distinguer des sous-groupes à l'intérieur d'une même étape PROC.
Exemple : on suppose que mavar a comme moyenne générale 10, comme moyenne sur la sous-population des hommes 14 et comme moyenne sur la sous-population des femmes 09. Admettons qu'on veuille faire un test d'égalité des moyennes. Le programme PROC TTEST ; VAR mavar ; CLASS sexe ; compare bien 09 à 14, tandis que le programme PROC TTEST ; VAR mavar ; BY sexe ; va essayer d'abord de comparer 09 à 09, puis de comparer 14 à 14.

L'instruction OUTPUT, qui permet de **stocker certains des résultats dans une table**.
La syntaxe est alors OUTPUT OUT=table_sortie *Keywords* ; on précise le nom de la table dans laquelle seront stockés les résultats des statistiques dont les mots-clés sont précisés (liste *Keywords*).
Notons qu'il existe de nombreuses options du type out= qui ont le même but mais sont adaptées à un cas précis.

On retrouvera aussi fréquemment deux instructions déjà vues en étape DATA :

- L'instruction WHERE sert à **sélectionner des observations** particulières pour notre traitement.
- L'instruction FORMAT sert à **appliquer un format à certaines des variables traitées**.

➢ *Exemple :*

```
1       PROC MEANS data=donnees ;
2         VAR var1 var2 ;
3         BY cat ;
4         WHERE cat NE 'sans reponse' ;
5         OUTPUT out=moyennes mean=moy1 moy2 ;
6       RUN ;
```

1 : Procédure MEANS qui s'applique à la table donnees de la librairie Work.
2 : Les variables étudiées sont var1 et var2.
3 : On calcule les stats sur les sous-populations définies par les modalités de cat.
4 : On exclut de l'étude les observations pour lesquelles cat vaut 'sans réponse'.
5 : On récupère dans une table moyennes la moyenne de chacune des variables sur chacune des sous-populations ; on appelle moy1 la moyenne de var1 et moy2 la moyenne de var2.

Partie II A

Procédures de manipulation des données

Les PROC IMPORT, EXPORT, CPORT permettent de faire de l'importation / exportation de données.

La PROC CONTENTS liste les propriétés d'une table.

La PROC PRINT édite les observations d'une table dans l'Output.

La PROC SORT trie une table.

La PROC TRANSPOSE transpose tout ou partie d'un tableau.

La PROC SQL permet d'utiliser les fonctionnalités particulières du langage SQL propre aux bases de données.

La PROC FORMAT crée un format, utile pour faire du regroupement de modalités.

La PROC RANK permet d'attribuer des rangs à des observations.

Quelques notions sur l'Importation / Exportation de données

Lorsque l'on dispose du module **SAS/WIZARD**, cela reste le plus simple moyen de faire de l'importation / exportation de données SAS depuis / vers les *formats texte, excel, access et dbase*.

SAS/WIZARD est accessible via les menus **File – Import data** et **File – Export data**. Il ne s'agit ni plus ni moins que d'un assistant implémentant les PROC IMPORT et PROC EXPORT de SAS.

Le lien avec d'autres logiciels statistiques tels que SPAD ou SPSS est moins évident. Dans de tels cas, on a recours au moteur **XPORT** pour transformer les données en un fichier transportable xpt.
Le programme suivant crée un fichier table.xpt stocké sous w:/sas à partir d'une table nommée table appartenant à la librairie base :

```
LIBNAME trans XPORT « w:\sas\table.xpt » ;
PROC COPY IN=base OUT=trans ;
SELECT table ;
RUN ;
```

SPAD peut lire directement les fichiers xpt.
Pour ouvrir la table sous SPSS, il faut taper dans une fenêtre de syntaxe la procédure suivante :

```
Get SAS DATA='w:\sas\table.xpt'.
Execute.
```

On peut via SPSS enregistrer une table au format por puis sous SAS utiliser la **PROC CONVERT** pour importer.
Le programme suivant crée une table SAS nommée table dans la librairie malib à partir du fichier table.por. On doit allouer un nom virtuel (ici : trans) à ce dernier grâce à l'instruction FILENAME.

```
FILENAME trans SPSS « w:\sas\base.por » ;
PROC CONVERT SPSS=trans OUT=malib.table ;
RUN ;
```

Les **PROC CPORT** et **PROC CIMPORT** sont utilisées pour transporter des fichiers SAS (tables, catalogues ou bibliothèques) d'un système d'exploitation à un autre ou d'une version de SAS à une autre. Un objet SAS codé avec la PROC CPORT ne peut être décodé qu'avec une PROC CIMPORT.

Connaître les propriétés d'une table avec la PROC CONTENTS

La première chose que l'on peut faire lorsqu'on reçoit des données à analyser, c'est de dresser la liste synthétique des caractéristiques de la table en question. La PROC CONTENTS est justement là pour ça.

Elle liste :

- Les propriétés générales de la table : nom, moteur, date de création, nombre de variables et d'observations…
- Les attributs des variables et leur position dans la table
- Les informations liées au tri de la table

➢ *Méthode alternative :*

- Dans la fenêtre Explorer, faire un clic droit sur la table concernée.
- Choisir « Properties ».

Une pop-up s'ouvre avec les propriétés de la table : apparaissent les General Properties (propriétés générales) de la table (qui correspondent au 1er tableau édité par la PROC CONTENTS), mais on peut aussi demander les Engine / Host Information (2ème tableau de la PROC) ou les Columns (informations relatives aux variables de la table, correspondant au 3ème tableau édité par la PROC).

```
PROC CONTENTS data=   ;
```

Options :

L'option short permet de n'éditer que la liste des variables.

L'option directory édite en plus les propriétés de la librairie à laquelle appartient la table (moteur, emplacement physique, liste des objets qu'elle contient avec leurs tailles).

Lecture de sortie :

```
                          The CONTENTS Procedure

Data Set Name: MALIB.BIDON                        Observations:         50
Member Type:   DATA                               Variables:            6
Engine:        V8                                 Indexes:              0
Created:       9:52 Monday, January 28, 2002      Observation Length:   40
Last Modified: 9:52 Monday, January 28, 2002      Deleted Observations: 0
Protection:                                       Compressed:           NO
Data Set Type:                                    Sorted:               NO
Label:

                  -----Engine/Host Dependent Information-----

                Data Set Page Size:         4096
                Number of Data Set Pages:   1
                First Data Page:            1
                Max Obs per Page:           101
                Obs in First Data Page:     50
                Number of Data Set Repairs: 0
                File Name:                  W:\SAS\bidon.sas7bdat
                Release Created:            8.0101M0
                Host Created:               WIN_NT

      -----Alphabetic List of Variables and Attributes-----

 #    Variable        Type    Len    Pos    Label

 6    distribution    Char     10     27    locale, régionale ou nationale
 1    siren           Char      8     16    identifiant SIRENE
 4    taille          Char      1     24    petite, moyenne ou grande entreprise
 2    va              Num       8      0    valeur ajoutée (en MF)
 3    ventes          Num       8      8    parts de marché
 5    zone            Char      2     25    implantation de l'entreprise
```

La table s'appelle bidon et est contenue dans la librairie malib. Elle est au format V8 et son emplacement physique est w:\sas\bidon.sas7bdat. Elle a été créée le 28 janvier 2002. Elle contient 6 variables et 50 observations et n'est pas triée.

Parmi les 6 variables de la table, 2 sont numériques :

- variable va, de label « valeur ajoutée (en MF) », de longueur 8, c'est la deuxième colonne
- variable ventes, de label « parts de marché », de longueur 8 aussi, troisième colonne

et 4 sont des variables caractères :

- variable siren, de longueur 8 et de label « identifiant SIRENE »
- variable taille, de longueur 1 et de label « petite, moyenne ou grande entreprise »
- variable zone, de longueur 2 et de label « implantation de l'entreprise »
- variable distribution, de longueur 10 et de label « entreprise locale, régionale ou nationale »

Imprimer une table dans la fenêtre Output avec la PROC PRINT

Pour voir le contenu d'une table, le plus simple reste encore de l'ouvrir en double-cliquant dessus. Lorsque c'est impossible, on peut utiliser le menu View – Table Editor, ou bien encore éditer la table dans l'output grâce à une PROC PRINT.

```
PROC PRINT data=  <options> ;
```

Les options :

L'option noobs supprime l'édition du numéro de l'observation.

Trier une table avec la PROC SORT

Pour trier une table, on utilise la PROC SORT. Cette procédure ne génère aucune sortie dans la fenêtre OUTPUT.

➢ *Méthode alternative :*

- Ouvrir la table par double clic.
- Passer en mode edit : menu Edit - Edit mode (ou l'icône Edit).
- Clic droit sur l'en-tête de colonne concerné.
- Choisir « Sort ».
- Puis on a le choix entre « Ascending » et « Descending ».
- Enfin on enregistre la table.

On peut aussi trier la table via le module SAS/INSIGHT (voir le paragraphe consacré dans la partie III B).

```
PROC SORT data=  <options> ;
  BY <descending> var ;
```

Les options :

L'option out= permet de spécifier une table de sortie différente de la table d'entrée : ainsi la table d'origine n'est pas modifiée, mais on dipose d'une nouvelle table qui, elle, est triée.

L'option noduprecs supprime les doublons.

Transposer un tableau avec la PROC TRANSPOSE

Dans son expression la plus simple, la PROC TRANSPOSE transpose une table, c'est-à-dire transforme les lignes en colonnes. La force de la PROC TRANSPOSE est de pouvoir ne transposer qu'une partie des observations.
La PROC TRANSPOSE ne génère aucune sortie dans l'OUTPUT.

```
PROC TRANSPOSE data=  <options>;
  VAR liste_de_variables;
  BY variable;
  COPY liste_de_variables;
  ID variable;
```

Les options :

L'option out= permet de définir une table en sortie différente de la table prise en entrée.
L'option name= permet de renommer la variable _name_ dans le tableau transposé (variable qui contient le nom de la ou des variables transposées).
L'option prefix= permet de définir un préfixe pour les noms des nouvelles variables du tableau.

Les instructions :

L'instruction VAR fournit la liste des variables qui vont être transposées. Ces variables n'existent plus dans le tableau en sortie, et leurs valeurs sont réparties dans les cases appropriées du nouveau tableau.
L'instruction BY permet de définir l'identifiant de l'individu. Chaque modalité de cet identifiant correspondra à une ligne du tableau final.
L'instruction ID permet de définir la variable dont les modalités doivent définir les nouvelles variables du tableau.
L'instruction COPY définit les variables qui sont recopiées telles quelles dans le tableau final.

Exemple :

Supposons que l'on dispose de la table notes suivante :

table notes

eleve	Matiere	note
tata	Math	15
tata	Français	10
titi	Math	14
titi	Français	18
toto	Math	8
toto	Français	12

Et que l'on souhaite obtenir la table notes2 suivante :

table notes2

eleve	note_math	note_francais
tata	15	10
titi	14	18
toto	8	12

Alors on écrit le programme suivant :

```
PROC SORT data=notes ; BY eleve ;
PROC TRANSPOSE data=notes out=notes2;
  VAR note;
  ID matiere;
  BY eleve;
RUN;
```

Ce programme fournit le résultat suivant :

table notes2

eleve	_NAME_	math	fran_ais
tata	note	15	10
titi	note	14	18
toto	note	8	12

On obtient presque ce que l'on souhaitait.
L'instruction BY eleve permet bien d'obtenir une ligne par élève.
L'instruction ID matiere définit bien la variable à transposer. La variable matiere n'existe plus, mais a été remplacée par autant de variables qu'il y avait de modalités de matiere. A noter que les caractères spéciaux (ici le ç) ne peuvent être traduits dans le nom de la nouvelle variable (ici il a été remplacé par un _).
Enfin l'instruction VAR note définit le contenu des nouvelles cases : dans mon tableau croisé eleve*matiere, je souhaite mettre la note qui correspond à ce profil.
Une colonne subsidiaire est créée, qui rappelle le nom de la variable que l'on a transposé.

Faire des jointures de tables avec la PROC SQL

Le langage SQL (Structured Query Language) est un langage de définition, de manipulation et de contrôle des données au sein d'une base de données.
On parle souvent à tort et à travers de bases de données. Le principe en est d'élaborer un système de tables reliées entre elles et organisées de manière à éviter au maximum la redondance d'informations et à faciliter la manipulation de données parfois très volumineuses.

SAS *n'est pas* un système de gestion de bases de données. En particulier, les tables de données sous SAS n'ont aucun lien entre elles. Mais bien qu'il n'y ait pas à proprement parler de bases de données sous SAS, il peut être pratique de recourir au langage SQL dès que l'on traite des données appartenant à des tables différentes.

La PROC SQL permet d'écrire des requêtes SQL sur des tables SAS. Il existe un outil « presse-bouton » qui permet de faire du SQL sans douleur : SAS Query (voir le paragraphe consacré dans la partie III B).

Le langage SQL est très riche, on propose ici de n'aborder que la syntaxe de requêtes simples.

Une PROC SQL n'a pas besoin de RUN; pour fonctionner. Par contre, on doit ajouter un QUIT; à la fin.

Fusions contrôlées et jointures :

Reprenons l'exemple de fusion de deux tables abordé dans la Partie I, paragraphe « Concaténation et fusion de tables ».

Le programme de fusion contrôlée :

```
PROC SORT data=a; BY nom;
PROC SORT data=b; BY nom;
DATA ab;
  MERGE a (rename=(note=note_a)) b (rename=(note=note_b));
  BY nom;
RUN;
```

Peut être remplacé par le programme SQL suivant :

```
PROC SQL;
 CREATE TABLE ab AS
  SELECT a.nom,a.note as note_a,b.note as note_b
  FROM a FULL JOIN b
  ON a.nom=b.nom;
QUIT;
```

➢ *Traduction du programme SQL :*

Je crée une table qui a pour nom ab, comme étant le résultat de la requête suivante :
Je sélectionne les variables : nom provenant de a, note provenant de a (que j'appelle note_a), note provenant de b (que j'appelle note_b)
A partir des tables a et b dont je fais une jointure complète
Cette jointure portant sur la condition que le nom provenant de a égale le nom provenant de b.

➢ *Les différents types de jointure en SQL :*

- La jointure complète permet de garder toutes les lignes, même celles qui n'appartiennent qu'à une des deux tables jointes. C'est ce type de jointure qui correspond à la fusion contrôlée en SAS, comme l'exemple précédent le montre.
- La jointure interne permet de ne garder que les individus appartenant aux deux tables jointes. C'est lorsqu'on souhaite réaliser ce type de fusion que la PROC SQL devient intéressante (voir l'exemple ci-dessous).
- Il existe aussi des jointures à gauche (on ne garde de la seconde table que les individus appartenant à la première table, mais on garde tous les individus de la première table) et à droite.

➢ *Exemple de jointure interne :*

Supposons que dans notre fusion précédente on n'ait souhaité conservé que les élèves appartenant à la fois à la table a et à la table b. L'étape DATA s'alourdit d'un test :

```
DATA ab;
  MERGE a (rename=(note=note_a) in=x) b (rename=(note=note_b) in=y);
  BY nom;
  IF x AND y THEN OUTPUT;
RUN;
```

Tandis que dans le programme SQL, il suffit de remplacer FULL JOIN par INNER JOIN :

```
PROC SQL;
 CREATE TABLE ab AS
  SELECT a.nom,a.note as note_a,b.note as note_b
  FROM a INNER JOIN b
  ON a.nom=b.nom;
QUIT;
```

- ***Quelques bonnes raisons de préférer SQL à MERGE :***

- Pas besoin de PROC SORT. En effet, contrairement aux fusions contrôlées, les jointures ne réclament pas que les tables à joindre soient précédemment triées.
- Pas besoin que la variable de contrôle porte le même nom dans les deux tables.
- On peut définir l'ordre dans lequel on veut que les variables apparaissent dans la table (c'est celui défini dans la clause SELECT), alors qu'avec la DATA, on aura forcément les variables de a *puis* les variables de b.
- Le langage est intuitif.

Un autre exemple d'utilisation de la PROC SQL

Supposons que l'on dispose de la table langues suivante :

table langues

nom	langue
tata	anglais
tata	allemand
tata	espagnol
titi	allemand
toto	anglais
toto	allemand
tutu	anglais
tutu	espagnol

Et que l'on souhaite avoir la liste des élèves ayant choisi exactement deux langues.

Sans la PROC SQL, il faudrait :

- Calculer le nombre de langues pour chacun des élèves (le plus simple est alors d'utiliser une PROC FREQ ; voir le paragraphe consacré dans la partie II B).
- Effectuer une étape DATA pour filtrer les seuls élèves ayant choisi deux langues.

Avec la PROC SQL, on écrit :

```
PROC SQL;
 CREATE TABLE qui2langues AS
  SELECT DISTINCT nom
  FROM langues
  GROUP BY nom HAVING count(*)=2 ;
QUIT;
```

- ***Traduction du programme SQL :***

Je crée une table de nom qui2langues comme étant le résultat de la requête suivante :
Je sélectionne les valeurs distinctes de nom
A partir de la table langues
Je groupe mes observations par nom et ne garde que les noms qui apparaissent 2 fois dans la table.

Créer un format avec la PROC FORMAT

Supposons que l'on dispose d'une variable numérique continue salaire et que l'on souhaite faire un traitement qui utilise les tranches de salaire. Trois possibilités s'offrent :

- Créer une nouvelle variable avec une étape DATA, qui à chaque observation associe la tranche de salaire de l'individu. Problème : Redondance d'information.
- La même chose, mais en supprimant la variable salaire : Perte d'information.
- La bonne solution consiste à créer un format.

Un format est un objet SAS, qui existe en propre et indépendemment de toute table, et qui associe des modalités caractère à des plages de valeur données (numériques ou caractères).

Un format se crée grâce à une PROC FORMAT. La procédure ne génère pas de sortie dans l'OUTPUT.

```
PROC FORMAT <options> ;
  VALUE nom_format plage1= « valeur1 » plage2= « valeur2 » … ;
```

Une fois la PROC FORMAT soumise, le format est créé pour toute la durée de la session.
On peut aussi stocker le format compilé dans un catalogue, afin de pouvoir l'utiliser une prochaine fois sans avoir à soumettre à nouveau le programme (voir la partie III C).

Le nom d'un format ne doit pas excéder 8 caractères et ne doit pas finir par un chiffre.
Si les plages de valeurs sont numériques, le format est dit numérique. Sinon, il est dit caractère, et son nom doit commencer par un $.
Les plages de valeurs peuvent s'écrire sous forme d'intervalle (par exemple : 0-100) ou sous forme de liste (par exemple : 0,1,2,5). Le signe < sert à exclure une borne de l'intervalle. Les plages doivent être des ensembles disjoints.

Exemple :

```
PROC FORMAT ;
  VALUE tranche
       low-<12500 = «pauvre»
       12500-30000 = «moyen»
       30000<-high = «riche» ;
RUN ;
```

Les options :

Pour éviter d'avoir à soumettre la PROC FORMAT à chaque session, on peut choisir de sauvegarder le format compilé dans un catalogue. Pour cela, il suffit de rajouter une option library=*nom_catalog* au moment de la création du format.

Si au contraire on dispose d'un catalogue de formats dont on ne connaît pas le code source, on peut visualiser leurs propriétés (plages des valeurs et modalités associées) en lançant une PROC FORMAT avec options library= et fmtlib, et sans instruction value :

`PROC FORMAT library=malib.formats fmtlib ; RUN ;` liste les formats contenus dans le catalogue formats.sas7bcat de la librarie malib.

Application du format :

Le format fonctionne comme une étiquette que l'on « colle » sur une variable le temps d'une procédure.
Supposons que l'on souhaite connaître le nombre moyen d'enfants dans les familles pauvres, dans les familles à revenus moyens et dans les familles riches. On va donc constituer des sous-groupes grâce à une instruction BY salaire ; encore faut-il lui préciser qu'à ce moment il doit lire les valeurs de salaire non comme des valeurs numériques mais comme une des modalités « pauvre », « moyen » ou « riche ». Pour cela, on applique le format à la variable salaire grâce à une instruction FORMAT :

```
FORMAT nom_variable nom_format. ;
```

```
PROC MEANS data=donnees ;
  FORMAT salaire tranche. ;
  VAR nbenfants ;
  BY salaire ;
RUN;
```

Attribuer des rangs aux observations avec la PROC RANK

La procédure PROC RANK recopie une table en y ajoutant une variable représentant le « rang » d'une observation par rapport à une variable donnée. La procédure ne génère pas de sortie dans l'output.

```
PROC RANK  data=    out=    <options> ;
      VAR variable ;
      RANKS nom_pour_nouvelle_variable ;
      < BY variable ; >
```

Les options :

L'option data= spécifie toujours la table sur laquelle on travaille.
L'option out= spécifie le nom de la table résultat.

L'option descending permet d'ordonner en sens inverse.

L'option ties= permet de spécifier comment traiter le cas des ex-aequo. Les choix possibles sont high, low et mean.

L'option percent permet de calculer des pourcentages cumulés : chaque rang est remplacé par le rang fois 100 rapporté au nombre d'observations non manquantes. Ainsi, lorsqu'une observation a un rang égale à 8, c'est que 8% des observations ont une valeur inférieure à celle de ladite observation.

L'option groups=n où $n \in \mathbb{N}^*$ permet de calculer des quantiles. Les observations sont regroupées selon leur rang en n groupes.

Les instructions :

L'instruction VAR permet de préciser selon quelle variable se fait le calcul de rang.
L'instruction RANKS permet de stocker le rang dans une nouvelle variable.

Exemple :

Soit la table jeu suivante :

joueur	points
A	15
B	10
C	23
D	14
E	22
F	9
G	10
H	7

Et le programme :

```
Proc rank data=jeu out=jeures descending ties=high ;
  Var points ;
  Ranks rang ;
Run;
```

Alors la table jeures contient :

joueur	points	rang
A	15	3
B	10	6
C	23	1
D	14	4
E	22	2
F	9	7
G	10	6

Les deux exemples suivants portent sur la table jeu, privée de sa dernière observation.

Le programme suivant :

```
Proc rank data=jeu out=jeures groups=3 ;
  Var points ;
  Ranks rang ;
Run;
```

Édite la table jeures suivante :

joueur	points	rang
A	15	1
B	10	0
C	23	2
D	14	1
E	22	2
F	9	0

Avec cela on voit par exemple que C fait partie des 2 meilleurs tandis que B et F sont les deux moins bons.

Le programme suivant :

```
Proc rank data=jeu out=jeures percent ;
  Var points ;
  Ranks rang ;
Run;
```

Édite la table jeures suivante :

joueur	points	rang
A	15	66,66666
B	10	33,33333
C	23	100
D	14	50
E	22	83,33333
F	9	16,66666

On lit par exemple que 1/3 des joueurs ont eu un score inférieur ou égal à celui de B, donc inférieur ou égal à 10.

Partie II B

Procédures de traitement statistique

Cette partie a pour but de présenter quelques procédures de traitement statistique parmi les plus connues, en rappelant leurs syntaxes avec les options et les instructions principales et en fournissant quelques interprétations de sorties.

Le but n'est pas de fournir une liste exhaustive des options et instructions, chose que vous trouverez sans peine dans l'aide du logiciel, ni de faire un exposé théorique des concepts statistiques qui reposent derrière ces procédures.

Cependant, quelques procédures, comme la PROC CORR ou la PROC REG, sont particulièrement détaillées.

La PROC FREQ étudie les variables qualitatives nominales (tableaux de fréquence, tableaux de contingence, tests du Chi^2…).

La PROC MEANS édite des statistiques descriptives pour des variables quantitatives continues (moyenne, écart-type, quartiles…).

La PROC TABULATE édite des tableaux de statistiques descriptives. Son intérêt consiste en la paramétrisation des tableaux.

La PROC UNIVARIATE est la procédure la plus complète de statistiques descriptives sur variables continues. Elle permet également de tracer des boîtes à moustaches et de faire des tests d'adéquation à une loi normale.

La PROC CORR calcule des corrélations entre variables numériques : soit entre des variables continues (corrélations de Pearson) soit entre des variables ordinales (Tau-b de Kendall).

La PROC TTEST teste l'égalité des moyennes entre deux sous-populations.

La PROC REG effectue des régressions linéaires multiples.

La PROC LOGISTIC effectue des régressions linéaires sur des variables catégorielles (qualitatives).

La PROC SYSLIN et la PROC MODEL estiment des modèles à plusieurs équations (systèmes linéaires pour la SYSLIN, non linéaires pour la MODEL).

Tableaux de fréquence et de contingence avec la PROC FREQ

Le propos de la procédure PROC FREQ est de faire des statistiques univariées ou bivariées sur des variables nominales. Elle permet donc de dresser des tableaux de fréquence et/ou des tableaux de contingence. C'est également dans cette procédure que l'on trouvera l'opportunité de faire des tests du χ^2.

```
PROC FREQ data=   <order=   > ;
      TABLES listes_variables </options> ;
      < BY variable ; >
      < WEIGHT variable ; >
```

Les options :

La plus intéressante est l'option order= , qui permet de choisir l'ordre dans lequel les modalités apparaissent. Retenons que order=freq trie le tableau de fréquence par ordre d'effectif décroissant, order=data le trie selon l'ordre d'apparition des modalités dans la table.

L'instruction TABLES et ses options :

L'instruction TABLES peut prendre deux formes :

- ✓ TABLES var1 var2 … varn ; édite les tableaux de fréquence des variables var_1 jusqu'à var_n.
- ✓ TABLES var1*…*varn ; édite pour chaque profil de (var_1,…,var_{n-2}) le tableau croisé (tableau de contingence) de var_{n-1} par var_n.

L'option out= permet d'enregistrer le résultat de l'instruction TABLES dans une table.

Les autres options peuvent être résumées dans les tableaux suivants :

Pour un tableau de contingence

	ce que la procédure édite	comment le supprimer	
par défaut	effectif de la case	nofreq	noprint
	poucentages	nopercent	
	pourcentages en ligne	norow	
	pourcentages en colonne	nocol	

	ce que la procédure peut éditer	comment le demander
en plus	effectifs théoriques	expected
	écarts entre effectif théorique et réel	deviation
	contributions à la distance du chi²	cellchi2
	test du chi² et statistiques dérivées du chi²	chisq

Pour un tableau de fréquence

ce que la procédure édite	comment le supprimer
effectif de la modalité	nofreq
pourcentage	nopercent
effectifs cumulés	nocum
pourcentages cumulés	

Lecture de sorties :

```
proc freq data=malib.bidon;
 tables zone taille*zone;
run;
```

The FREQ Procedure

zone	Frequency	Percent	Cumulative Frequency	Cumulative Percent
IF	4	8.00	4	8.00
NE	9	18.00	13	26.00
NW	2	4.00	15	30.00
SE	15	30.00	30	60.00
SW	20	40.00	50	100.00

Table of taille by zone

taille zone

Frequency / Percent / Row Pct / Col Pct	IF	NE	NW	SE	SW	Total
1	0	4	0	11	18	33
	0.00	8.00	0.00	22.00	36.00	66.00
	0.00	12.12	0.00	33.33	54.55	
	0.00	44.44	0.00	73.33	90.00	
2	1	4	2	4	2	13
	2.00	8.00	4.00	8.00	4.00	26.00
	7.69	30.77	15.38	30.77	15.38	
	25.00	44.44	100.00	26.67	10.00	
3	3	1	0	0	0	4
	6.00	2.00	0.00	0.00	0.00	8.00
	75.00	25.00	0.00	0.00	0.00	
	75.00	11.11	0.00	0.00	0.00	
Total	4	9	2	15	20	50
	8.00	18.00	4.00	30.00	40.00	100.00

Exemple de commentaire :
40% des entreprises du secteur Bidon sont installées dans le sud ouest, ce qui représente 20 firmes. 90% de ces firmes sont des petites entreprises et 10% sont des PME. Les PME se retrouvent en effet majoritairement dans l'est, le nord est et le sud est regroupant chacun plus de 30% des entreprises concernées. Les grandes entreprises sont rares dans le secteur, puisqu'elles n'en représentent que 8%.

```
proc freq data=malib.bidon ;
 tables distribution*taille /chisq cellchi2 norow nocol;
 where distribution NE 'nationale';
run;
```

The FREQ Procedure

Table of distribution by taille

distribution	taille			
Frequency Cell Chi-Square Percent	1	2	3	Total
locale	33 0.7857 67.35	9 0.4121 18.37	0 2.5714 0.00	42 85.71
régionale	0 4.7143 0.00	4 2.4725 8.16	3 15.429 6.12	7 14.29
Total	33 67.35	13 26.53	3 6.12	49 100.00

Statistics for Table of distribution by taille

Statistic	DF	Value	Prob
Chi-Square	2	26.3846	<.0001
Likelihood Ratio Chi-Square	2	24.1431	<.0001
Mantel-Haenszel Chi-Square	1	24.0833	<.0001
Phi Coefficient		0.7338	
Contingency Coefficient		0.5916	
Cramer's V		0.7338	

WARNING: 67% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 49

La ligne Chi-Square du tableau généré par l'option chisq nous indique la valeur de la distance du χ^2 et la p-value associée. Ici on rejetterait l'hypothèse nulle d'indépendance des variables. Cela dit, on remarque que la plus grosse contribution à la distance provient des trois grandes entreprises à distribution régionale du secteur. Bien que l'existence d'une corrélation semble évidente à l'œil, on peut s'interroger sur la validité du test du χ^2 sur un échantillon aussi petit. D'ailleurs, SAS nous met en garde !

Statistiques descriptives quantitatives avec la PROC MEANS

Cette procédure est la première qu'il faut connaître lorsqu'on souhaite effectuer des statistiques descriptives élémentaires sur des variables quantitatives. Par défaut, elle calcule le nombre d'observations non manquantes, la moyenne, l'écart-type, la valeur minimum et la valeur maximum de toutes les variables numérique de la table (ou des variables indiquées par l'instruction VAR).

Avec les options appropriées, on peut demander un grand nombre de statistiques : somme, médiane, variance, skewness, kurtosis, quartiles, premier et dernier centile, premier et dernier décile, etc. Lorsqu'une (ou plusieurs) de ces options est spécifiée, cela annule l'édition par défaut. Il faudra donc expliciter toutes les statistiques que l'on souhaite obtenir.

```
PROC MEANS data=   <options> ;
      VAR liste_variables_quantitatives ;
      < BY variable ; >
      < CLASS variable ; >
      < FREQ variable ; >
      < WEIGHT variable ; >
      < ID variable ; >
      < OUTPUT <OUT=nom_table>
            <mot_cle_de_la_statistique=nom_stat_dans_la_table_sortie>
            <idem avec autres statistiques> ; >
```

Les options :

Les options les plus intéressantes sont celles qui permettent de choisir les statistiques à éditer :

N	Effectif
NMISS	Nombre de valeurs manquantes
MIN	Minimum
MAX	Maximum
RANGE	Plage des valeurs = MAX – MIN
SUMWGT	Somme des poids
SUM	Somme
MEAN	Moyenne
STD	Ecart-type
STDERR	Standard Error of Mean
KURTOSIS	Coefficient d'aplatissement
SKEWNESS	Coefficient d'asymétrie
USS	Somme des carrés
CSS	Somme des carrés des écarts à la moyenne
VAR	Variance
CV	Coefficient de variation
T	Valeur de la statistique de Student pour le test (H0) : la moyenne est nulle
PROBT	P-value associée au test précédent
MEDIAN	Médiane
QRANGE	Distance interquartile = Q3 - Q1
Q1 et Q3	Premier et troisième quartiles
P1 P5 P10 P90 P95 P99	Centiles

Remarque : Ces statistiques sont éditées par défaut par la PROC UNIVARIATE (voir paragraphe consacré).

Les instructions :

L'instruction VAR permet de préciser sur quelle(s) variable(s) on travaille. Si on omet l'instruction, la PROC MEANS prend en compte toutes les variables numériques de la table spécifiée.

Les instructions BY et CLASS permettent de faire des sous-groupes. L'instruction BY commande l'exécution de la procédure sur chacunes des sous-populations qu'elle définit. Elle requiert que la table soit triée. L'instruction CLASS regroupe les observations qui ont un certain profil. La procédure n'est exécutée qu'une seule fois et il n'est pas nécessaire que la table soit triée. De plus, la présentation des résultats est plus synthétique qu'avec BY. Mais les résultats édités sont les mêmes.

Lorsqu'une variable apparaît dans l'instruction FREQ, tout se passe comme si chaque observation i apparaissait v(i) fois dans la table, où v(i) est la valeur en i de la variable désignée par l'instruction FREQ.

L'instruction WEIGHT permet de définir une variable qui servira à pondérer les observations avant le calcul des corrélations.

L'instruction ID spécifie une variable qui sert d'identifiant pour les observations.

L'instruction OUTPUT sert à récupérer certaines des statistiques dans une table dont on spécifie le nom après OUT= . La liste des statistiques que l'on souhaite enregistrer dans cette table est spécifiée ensuite. On peut éventuellement décider de renommer ces variables, car le nom généré automatiquement par SAS n'est pas très clair.
Admettons que l'on veuille récupérer la moyenne et l'écart-type des trois variables qui ont été désignées dans l'instruction VAR par VAR v1 v2 v3 ; dans une table nommée resultat.

Alors on écrit :

```
OUTPUT  OUT=resultat  MEAN=moy1 moy2 moy3  STD=et1 et2 et3  ;
```

Alors moy1 désigne la moyenne de v1, moy2 la moyenne de v2,…

Lecture de sorties :

```
proc means data=malib.bidon ;
run ;
```

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
va	va (en MF)	50	1562.42	3154.38	139.0000000	17842.00
ventes		50	0.0200000	0.0433028	0.0020000	0.2630000

```
proc means data=malib.bidon sum ;
 var ventes ;
 class zone distribution ;
run ;
```

Analysis Variable : ventes

zone	distribution	N Obs	Sum
IF	nationale	1	0.2630000
	régionale	3	0.1960000
NE	locale	8	0.0190000
	régionale	1	0.0740000
NW	régionale	2	0.1890000
SE	locale	14	0.1740000
	régionale	1	0.0100000
SW	locale	20	0.0750000

La procédure calcule la somme des parts de marché détenues par les entreprises correspondant à un certain profil (zone ; distribution). Par exemple, les entreprises locales implantées dans le sud-est représentent 17% du marché.

Des tableaux de statistiques descriptives avec la PROC TABULATE

Cette procédure est utile pour dresser des tableaux de statistiques descriptives. Les statistiques que la PROC TABULATE peut éditer sont sensiblement les mêmes que dans la PROC MEANS. La force de cette procédure résidant dans la présentation des résultats.

```
PROC TABULATE  <data= > <options> ;
  CLASS variables ;
  < CLASSLEV variables / style= ; >
  < KEYLABEL keyword='description' ; >
  < KEYWORD keywords / style= ; >
  TABLE <<page_expression>, row expression>, column expression ;
  VAR variables ;
```

Les instructions :

- *Pour le calcul des statistiques descriptives :*

L'instruction CLASS permet de définir les variables catégorielles qui seront utilisées dans la proc, soit pour être étudiées (tableaux de fréquence et de contingence) soit pour constituer des sous-groupes.

L'instruction VAR permet, comme d'habitude, de préciser les variables (numériques) sur lesquelles on souhaite travailler.

L'instruction TABLE définit à la fois ce qui est calculé et la manière dont les résultats sont présentés dans le tableau. On doit au moins définir les éléments qui constituent les colonnes du tableau. Il s'agira d'une combinaison de noms de variables et de mots-clés de statistiques, reliés par des opérateurs :

- L'opérateur 'espace' est la concaténation des éléments
- L'opérateur '*' réalise le croisement des éléments
- L'opérateur ',' sépare ce qui sera en ligne de ce qui sera en colonne.

Voir page d'après la liste des mots-clés.

Toutes les variables qualitatives invoquées doivent être définies au préalable dans l'instruction CLASS. Toutes les variables quantitatives invoquées doivent être répertoriées dans l'instruction VAR.

Exemples :

`CLASS sexe ;` `TABLE sexe ;`	Edite un tableau contenant les effectifs des modalités de sexe où les modalités sont les colonnes du tableau.
`CLASS sexe ;` `TABLE (n pctn),sexe ;`	Edite un tableau à deux lignes contenant les effectifs et pourcentages des modalités de sexe (les colonnes correspondent aux modalités de sexe).
`CLASS sexe diplome ;` `TABLE sexe*diplome ;`	Edite un tableau à une ligne (l'effectif) dont chaque colonne (modalités de sexe) sont subdivisées selon les modalités de diplome.
`TABLE salaire ;` `VAR salaire ;`	Edite un tableau à une colonne (le salaire) et une ligne (somme de la variable salaire).
`TABLE (sum mean std),salaire ;` `VAR salaire ;`	Édite un tableau à 3 lignes (la somme, la moyenne et l'écart-type) et 1 colonne (la variable salaire).

`CLASS sexe ;`
`TABLE (sum mean),salaire*sexe ;`
`VAR salaire ;`
	Edite un tableau à 2 lignes (la somme et la moyenne) et 1 colonne pour salaire, laquelle est subdivisée selon les modalités de sexe. Sont donc calculées la somme et la moyenne des salaires sur les sous-populations des hommes et des femmes.

- *Pour la présentation des résultats :*

L'instruction CLASSLEV permet d'affecter un style aux variables de groupe.
L'instruction KEYWORD permet d'affecter un style aux titres des statistiques éditées.

Note : Ces deux instructions utilisent l'option style= . Voir le paragraphe suivant pour plus de détails sur cette option.

L'instruction KEYLABEL affecte un libellé à une statistique.

- *Autres instructions :*

Les instructions BY, FREQ et WEIGHT sont également utilisables dans cette procédure (voir leur description dans le paragraphe sur la PROC MEANS).

Les options :

L'option classdata= permet de préciser le nom d'une table contenant les variables qui serviront à constituer les sous-groupes.
L'option exclusive élimine de l'analyse les profils qui ne figurent pas dans la table déclarée après classdata= .

L'option order= ordonne les profils selon la méthode spécifiée : data (ordre d'apparition dans la table) et freq (ordre décroissant de l'effectif des profils) sont les plus utiles.

L'option style= définit la mise en forme des cellules du tableau. L'option style=parent indique que les cellules héritent de la mise en forme du titre de la colonne. Sinon on définit les paramètres entre […]. On se réfèrera à l'aide en ligne de SAS pour une liste exhaustive des paramètres. Citons-en quelques uns :

Background	couleur de fond des cellules
Foreground	couleur du texte
Bordercolor	couleur de la bordure du tableau
Cell_height	hauteur des cellules
Cell_width	largeur des cellules
Font_face	police
Font_size	taille de la police

Exemple d'utilisation : style=[background=red]

Note : la mise en forme n'est visible que dans le fichier exporté par ODS (voir partie III C).

Liste des mots-clés statistiques :

N	Effectif
NMISS	Nombre de valeurs manquantes
PCTN	Pourcentage de l'effectif total
MIN	Minimum
MAX	Maximum
RANGE	Plage des valeurs = MAX – MIN
SUMWGT	Somme des poids
SUM	Somme
PCTSUM	Pourcentage de la somme
MEAN	Moyenne
STD	Ecart-type
STDERR	Standard Error of Mean
USS	Somme des carrés
CSS	Somme des carrés des écarts à la moyenne
VAR	Variance
CV	Coefficient de variation
T	Valeur de la statistique de Student pour le test (H0) : la moyenne est nulle
PROBT	P-value associée au test précédent
MEDIAN	Médiane
QRANGE	Distance interquartile
Q1 et Q3	Premier et troisième quartiles
P1 P5 P10 P90 P95 P99	Centiles

Lecture de sorties :

```
ods rtf file="w:/sas/cours/tabulate.rtf";
PROC TABULATE data=malib.bidon
            style=[font_size=2 font_face=verdana foreground=blue];
  CLASS zone distribution;
  CLASSLEV zone distribution
            / style=[background=white font_size=2 font_face=verdana];
  KEYLABEL pctn='percent' n='effectif' mean='moyenne' std='éct-type';
  TABLE (mean std),va*zone;
  TABLE zone,distribution*(n pctn);
  VAR va;
RUN;
ods rtf close;
```

On obtient la sortie .rtf suivante :

	va (en MF) zone				
	IF	NE	NW	SE	SW
moyenne	8279.50	790.89	8315.50	1003.80	309.85
écart-type	6876.15	1784.26	904.39	669.40	51.23

	distribution					
	locale		nationale		régionale	
	effectif	percent	effectif	percent	effectif	percent
zone						
IF	.	.	1	2.00	3	6.00
NE	8	16.00	.	.	1	2.00
NW	.	.	.	.	2	4.00
SE	14	28.00	.	.	1	2.00
SW	20	40.00	.	.	.	.

L'instruction `TABLE (mean std),va*zone;` calcule la moyenne et l'écart-type (deux lignes dans le tableau) de la valeur ajoutée sur chacune des sous-populations zone=IF, zone=NE, zone=NW, zone=SE et zone=SW (modalités en colonnes).

L'instruction `TABLE zone,distribution*(n pctn);` calcule l'effectif et le pourcentage de l'effectif total de chacun des profils de (zone, distribution) recensés dans la table. Les modalités de zone sont en ligne. Les modalités de distribution et les statistiques sont en colonne.

L'instruction `KEYLABEL pctn='percent' n='effectif' mean='moyenne' std='écart-type';` permet aux noms des statistiques d'être clairs.

Les résultats apparaissent en Verdana bleu de taille 2 (9 pt) grâce à l'option `style=[font_size=2 font_face=verdana foreground=blue]` de la PROC.
Les cases portant le titre des modalités ont un fond blanc grâce à l'instruction `CLASSLEV zone distribution / style=[background=white font_size=2 font_face=verdana];`

Les tableaux édités dans la fenêtre Output sont les mêmes, la mise en forme (couleur, police…) en moins.

Statistiques univariées et distributions avec la PROC UNIVARIATE

La procédure PROC UNIVARIATE est fort utile pour « déblayer le terrain » avant de se lancer dans des tests statistiques plus élaborés.

Par défaut, la sortie générée par la PROC UNIVARIATE comporte cinq blocs :

1. Calcul des statistiques descriptives suivantes : nombre d'observations, moyenne, écart-type, skewness (coefficient d'asymétrie), somme des carrés, coefficient de variation, somme des poids, somme de la variable, variance, kurtosis (coefficient d'aplatissement), somme des carrés des écarts à la moyenne, écart-type de la moyenne.
2. Paramètres élémentaires de distribution : moyenne, médiane, mode, écart-type, variance, écart entre maximum et minimum, et distance interquartile.
3. Tests pour (H0) : la moyenne est nulle (3 tests différents sont réalisés).
4. Quantiles essentiels.
5. Observations extrêmes (les 5 plus basses et les 5 plus élevées).

La PROC UNIVARIATE peut également réaliser un test de normalité ainsi que des graphiques de distribution : box plot (boîte à moustaches), diagramme stem and leaf et graphe d'ajustement à une loi normale.

C'est enfin la reine des quantiles, puisqu'elle peut calculer tous les centiles, il suffit de passer commande !

```
PROC UNIVARIATE  data=   <options> ;
   VAR liste_variables ;
   < PROBPLOT variable / options ; >
   < BY liste_variables ; >
   < FREQ variable ; >
   < WEIGHT variable ; >
   < ID liste_variables ; >
   < OUTPUT <OUT=nom_table>
            <mot_cle_de_la_statistique=nom_stat_dans_la_table_sortie>
            <idem avec autres statistiques>
            <PCTLPTS=liste_centiles  PCTLPRE=liste_préfixes_centiles>
            < PCTLNAMES=liste_suffixes_pour_centiles> ; >
```

Les options :

Les deux options les plus fréquemment utilisées sont l'option normal et l'option plots.

- L'option normal produit un test d'adéquation à une loi normale de la distribution de la (des) variable(s) indiquée(s) dans l'instruction VAR.

- L'option plots réalise les graphiques de distribution dont on a déjà parlé.

L'option freq qui génère l'édition de tableaux recensant les modalités de la (des) variable(s) indiquée(s) dans l'instruction VAR, ainsi que les effectifs, fréquences et fréquences cumulées desdites modalités.

L'option mu0= permet de définir la valeur de référence pour l'hypothèse nulle du test de location (par défaut, cette valeur est égale à 0, ce qui signifie que l'hypothèse nulle du test est : la moyenne est nulle).

L'option nextrobs= définit le nombre d'observations extrêmes que l'on souhaite éditer (par défaut ce paramètre vaut 5, ce qui signifie que les 5 plus basses et les 5 plus élevées des observations sont éditées).

Les instructions :

Les instructions VAR, BY, WEIGHT, FREQ et ID fonctionnent comme pour les autres procédures qui les utilisent. On peut se référer à l'explication donnée dans le cadre de la PROC MEANS.

L'instruction PROBPLOT permet de tracer la courbe de distribution de la variable indiquée, en la superposant à une courbe de distribution choisie parmi les lois classiques.
Ce choix (facultatif) s'effectue grâce à l'option que l'on indique derrière l'instruction PROBPLOT. Par exemple, pour tracer la courbe d'une loi normale, on utilise l'option normal. Cette option comporte comme sous-options (indiquées entre parenthèses) : mu= et sigma= qui précisent les paramètres de la loi ; mu=est (resp. sigma=est) indique que la moyenne (resp. l'écart-type) considérée est celle (celui) de l'échantillon.
On peut également choisir de tracer la courbe d'une loi bêta, gamma, exponentielle, lognormale, ou Weibull. On se réfèrera à l'aide de SAS pour le détail de ces options.
Les autres options ont trait à la présentation des courbes. Là encore, on se réfèrera à l'aide de SAS.

➢ *Calcul de centiles dans l'instruction OUTPUT :*

L'instruction OUTPUT sert toujours à récupérer certaines des statistiques dans une table dont on spécifie le nom après OUT= . La liste des statistiques que l'on souhaite enregistrer dans cette table est spécifiée ensuite.
C'est aussi dans cette instruction que l'on peut demander des centiles. Il suffit d'en donner la liste après le mot-clé PCTLPTS= . Il est nécessaire de préciser un ou des préfixes pour que SAS sache comment les nommer. On les précise après PCTLPRE= . Si on ne précise qu'un préfixe, il est le même pour tous les centiles demandés. Sinon, on peut en préciser autant qu'il y a de centiles : alors le premier préfixe sert au premier des centiles demandés, et ainsi de suite…
On peut ne pas vouloir récupérer ces quantiles dans une table. Il suffit pour cela de ne pas spécifier la commande OUT= . Notons qu'il est aussi possible de choisir d'autres suffixes que ceux par défaut (qui sont les numéros des centiles) grâce à la commande PCTLNAMES= .

La sortie standard et les mots-clés associés à chaque statistique éditée :

Ci dessous on figure l'allure d'une sortie standard de PROC UNIVARIATE, dans laquelle on a remplacé les valeurs des statistiques par les mots clés qui servent à désigner chacune de ces statistiques dans la liste de variables de l'instruction OUTPUT.

The UNIVARIATE Procedure
Variable : ...

Moments

N	N	Sum Weights	SUMWGT
Mean	MEAN	Sum Observations	SUM
Std Deviation	STD	Variance	VAR
Skewness	SKEWNESS	Kurtosis	KURTOSIS
Uncorrected SS	USS	Corrected SS	CSS
Coeff Variation	CV	Std Error Mean	STDMEAN

Basic Statistical Measures

Location		Variability	
Mean	MEAN	Std Deviation	STD
Median	MEDIAN	Variance	VAR
Mode	MODE	Range	RANGE
		Interquartile range	QRANGE

Tests for Location: Mu0=0.00

Test Statistic		Value	p-value	
Student's t	T	T	Pr > | t |	PROBT
Sign	M	MSIGN	Pr >= | M |	PROBM
Signed Rank	S	SIGNRANK	Pr >= | S |	PROBS

Quantiles (Definition 5)

Quantile	Estimate
100% Max	MAX
99%	P99
95%	P95
90%	P90
75% Q3	Q3
50% Med	MEDIAN
25% Q1	Q1
10%	P10
5%	P5
1%	P1
0% Min	MIN

Extreme Observations

------Lowest------		------Highest------	
Value	Obs	Value	Obs

Outre ces statistiques, il est également possible de demander dans l'instruction OUTPUT :

NMISS	nombre d'observations ayant une valeur manquante
NOBS	nombre total d'observations
NORMAL	statistique du test de normalité
PROBN	niveau de significativité du test de normalité

Exemple :

```
PROC UNIVARIATE plot normal ;
  VAR salaire ;  BY sexe ;
RUN ;
```

---------- sexe=F ----------

The UNIVARIATE Procedure
Variable: salaire (salaire en euros)

Moments

N	16	Sum Weights	16
Mean	1589	Sum Observations	25424
Std Deviation	714.659546	Variance	510738.267
Skewness	2.68443397	Kurtosis	8.59227515
Uncorrected SS	48059810	Corrected SS	7661074
Coeff Variation	44.9754277	Std Error Mean	178.664886

Basic Statistical Measures

Location		Variability	
Mean	1589.000	Std Deviation	714.65955
Median	1379.000	Variance	510738
Mode	1379.000	Range	2942
		Interquartile Range	694.00000

Tests for Location: Mu0=0

Test	-Statistic-		-----p Value------	
Student's t	t	8.893745	Pr > |t|	<.0001
Sign	M	8	Pr >= |M|	<.0001
Signed Rank	S	68	Pr >= |S|	<.0001

Tests for Normality

Test	--Statistic---		-----p Value------	
Shapiro-Wilk	W	0.685914	Pr < W	0.0001
Kolmogorov-Smirnov	D	0.282596	Pr > D	<0.0100
Cramer-von Mises	W-Sq	0.270147	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	1.643833	Pr > A-Sq	<0.0050

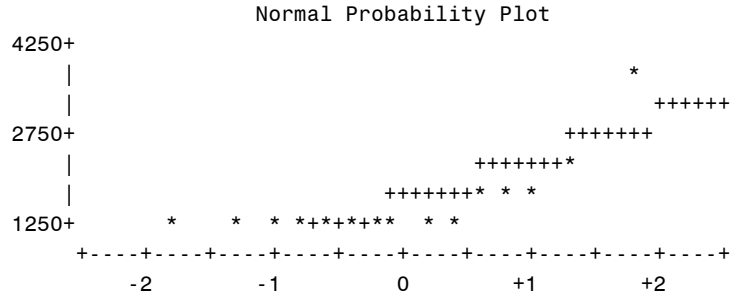
Quantiles (Definition 5)

Quantile	Estimate
100% Max	3963.0
99%	3963.0
95%	3963.0
90%	2088.0
75% Q3	1859.5
50% Median	1379.0
25% Q1	1165.5
10%	1033.0
5%	1021.0
1%	1021.0
0% Min	1021.0

Extreme Observations

----Lowest----		----Highest---	
Value	Obs	Value	Obs
1021	15	1844	8
1033	13	1875	4
1067	16	1905	3
1097	9	2088	2
1234	14	3963	1

```
Stem Leaf                     #             Boxplot
   4 0                        1                *
   3
   3
   2
   2 1                        1                |
   1 899                      3             +--+--+
   1 00112444444             11             *-----*
     ----+----+----+----+
  Multiply Stem.Leaf by 10**+3
```



---------- sexe=H ----------

The UNIVARIATE Procedure
Variable: salaire (salaire en euros)

Moments

N	15	Sum Weights	15
Mean	2264.33333	Sum Observations	33965
Std Deviation	1395.26227	Variance	1946756.81
Skewness	1.74834242	Kurtosis	3.26071832
Uncorrected SS	104162677	Corrected SS	27254595.3
Coeff Variation	61.6191199	Std Error Mean	360.25517

Basic Statistical Measures

Location		Variability	
Mean	2264.333	Std Deviation	1395
Median	1951.000	Variance	1946757
Mode	.	Range	5129
		Interquartile Range	1160

Tests for Location: Mu0=0

Test	-Statistic-		-----p Value------	
Student's t	t	6.28536	Pr > |t|	<.0001
Sign	M	7.5	Pr >= |M|	<.0001
Signed Rank	S	60	Pr >= |S|	<.0001

Tests for Normality

Test	--Statistic---		-----p Value------	
Shapiro-Wilk	W	0.814436	Pr < W	0.0057
Kolmogorov-Smirnov	D	0.230963	Pr > D	0.0300
Cramer-von Mises	W-Sq	0.165237	Pr > W-Sq	0.0136
Anderson-Darling	A-Sq	0.983563	Pr > A-Sq	0.0097

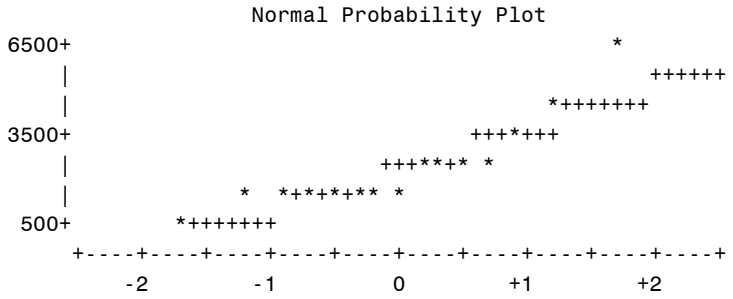
Quantiles (Definition 5)

Quantile	Estimate
100% Max	6097
99%	6097
95%	6097
90%	4268
75% Q3	2507
50% Median	1951
25% Q1	1347
10%	1036
5%	968
1%	968
0% Min	968

Extreme Observations

----Lowest----		----Highest---	
Value	Obs	Value	Obs
968	15	2286	4
1036	13	2507	6
1082	14	3353	3
1347	10	4268	2
1353	11	6097	1

```
Stem Leaf                     #             Boxplot
   6 1                        1                *
   5
   4 3                        1                0
   3 4                        1                |
   2 02235                    5               +--+--+
   1 0013449                  7               +-----+
   0
     ----+----+----+----+
  Multiply Stem.Leaf by 10**+3
```

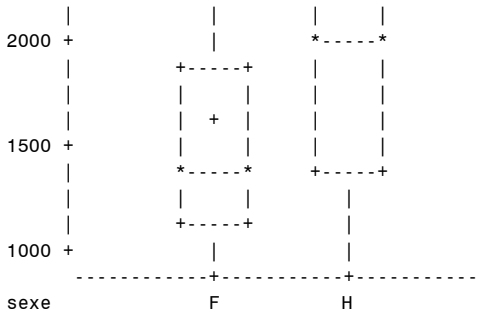


Les deux analyses ci-dessus correspondent au calcul de statistiques univariées sur les deux sous-populations définies par les modalités de la variable sexe (instruction BY) : sous-population masculine et sous-population féminine. Outre la sortie standard, un test de normalité a été édité : vu que la taille de notre échantillon est assez petite, on se réfère au test de Shapiro-Wilk. La p-value étant très inférieure à 0,05 dans les deux cas, on rejette à 5% l'hypothèse d'adéquation à une loi normale des deux distributions de salaire considérées.
Ci-après l'édition des boîtes à moustaches générées par l'option PLOT.

```
        The UNIVARIATE Procedure
 Variable:  salaire  (salaire en euros)

             Schematic Plots

      |
      |                      *
 6000 +
      |
      |
      |
 5500 +
      |
      |
      |
 5000 +
      |
      |
      |
 4500 +
      |
      |                      0
      |
 4000 +           *
      |
      |
      |
 3500 +
      |                      |
      |                      |
      |                      |
 3000 +                      |
      |                      |
      |                      |
      |                      |
 2500 +                   +-----+
      |                   |     |
      |                   |  +  |
```



On préfèrera éditer les boxplots avec une procédure adaptée, soit la PROC BOXPLOT (voir le paragraphe consacré dans la partie II C).

Les corrélations avec la PROC CORR

La procédure PROC CORR permet de calculer des coefficients de corrélation entre variables numériques.

Par défaut, la procédure édite la matrice des corrélations linéaires :

$$r_{XY} = \frac{\mathrm{cov}(x, y)}{\sigma_X . \sigma_Y}$$

Alors il existe une relation linéaire entre *X* et *Y* si et seulement si $| r_{XY} | = 1$.
La procédure édite également la p-value relative au test de significativité de chaque coefficient. Si cette valeur est inférieure à 0,05 on peut conclure que le coefficient est significativement différent de zéro au seuil de 5%.

➢ *Autres mesures de la dépendance entre variables quantitatives :*

On peut également demander l'édition du alpha de Cronbach, qui, en quelque sorte, mesure la « cohérence interne » de la somme des variables considérées dans la procédure. La valeur maximale de α est 1. Si α vaut 1, c'est que toutes les variables sont liées linéairement deux à deux.

La PROC CORR offre une autre possibilité pour mesurer la dépendance entre deux variables quantitatives *X* et *Y* : la statistique D de Hoeffding. On n'explicite pas le détail de la statistique ici, il suffit pour le moment de savoir que plus D est proche de 1, plus il existe une dépendance forte entre *X* et *Y*.

Il est également possible de demander les coefficients de corrélation partielle (corrélation corrigée de l'influence de certaines variables), bien utiles pour éliminer les effets de structure. Le coefficient de corrélation partielle entre *X* et *Y* corrigé de l'influence de *Z1,…,Zp* est égal au coefficient de corrélation linéaire entre les résidus e_X et e_Y des régressions de *X* et *Y* sur *Z1,…,Zp*.

➢ *Le tau-b de Kendall et le coefficient de corrélation des rangs de Spearman :*

D'autre part, il est possible d'obtenir le coefficient τ_b de Kendall ou le coefficient de corrélation des rangs de Spearman, qui mesurent la corrélation entre deux variables ordinales :

$$\tau_b(X, Y) = \frac{C - D}{\sqrt{(n^2 - E_X)(n^2 - E_Y)}}$$

où
n^2 est le nombre total de paires d'individus (i ; j)
C est le nombre de paires (i ; j) concordantes , c'est-à-dire telles que i et j sont classés dans le même ordre pour les variables *X* et *Y*
D est le nombre de paires discordantes
E_X est le nombre d'ex-aequo au moins pour *X*, c'est-à-dire *Xi = Xj*
E_Y est le nombre d'ex-aequo au moins pour *Y*

Si $| \tau_b |$ = 1, il existe alors une relation biunivoque entre *X* et *Y*. On pourra dire que *X* est une fonction strictement (dé)croissante de *Y* et vice versa.

Pour le coefficient de corrélation des rangs de Spearman, on commence par calculer *rangX* et *rangY* ; *rangX* vaut 1 pour la plus grande valeur de *X*, 2 pour la deuxième plus grande valeur de *X*, etc…
Et alors ρ_s est le coefficient de corrélation linéaire entre *rangX* et *rangY*.

Remarque : Comme il s'agit d'une procédure qui traite des variables numériques, on ne peut pas lui demander de traiter des variables caractères ! Les coefficients τ_b et ρ_s seront édités sur des variables numériques auxquelles on aura appliqué un format.

Tous les autres coefficients relatifs à la corrélation entre variables qualitatives s'obtiennent avec la PROC FREQ, dont la raison d'être est justement le traitement statistique des variables qualitatives.

```
PROC CORR   data=   <options> ;
   VAR liste_variables ;
   < WITH liste_variables ; >
   < BY liste_variables ; >
   < FREQ variable ; >
   < PARTIAL liste_variables ; >
   < WEIGHT variable ; >
```

Les options : *(voir tableau récapitulatif page suivante)*

Par défaut la procédure calcule les coefficients de corrélation linéaire dits de Pearson.

Pour obtenir le τ_b de Kendall, il faut préciser l'option kendall.
Pour obtenir le coefficient de corrélation des rangs de Spearman, il faut préciser l'option spearman.
Enfin pour obtenir le D de Hoeffding, il faut préciser l'option hoeffding.
Notons qu'alors la matrice des corrélations linéaires n'est plus éditée.
Si on souhaite l'obtenir tout de même, il faut préciser l'option pearson.

On peut récupérer les corrélations linéaires dans une table grâce à l'option outp= , et pareil pour les autres coefficients grâce à outk= , outs= et outh= .

Parmi les options qui s'utilisent avec l'option PEARSON, mentionnons l'option cov qui édite la matrice des covariances et l'option nocorr qui supprime le calcul des corrélations linéaires.

Parmi les autres options, la plus fréquemment utilisée est l'option rank, qui permet d'éditer les corrélations par ordre décroissant. Si le nombre de variables croisées est important est que l'on ne veut garder dans le listing que les 5 meilleures corrélations (par exemple), on précisera l'option best=5.

Les instructions :

L'instruction VAR permet de préciser les variables auxquelles on s'intéresse. Elles doivent être numériques.

Par défaut, la matrice des corrélations croise toutes les variables définies dans VAR deux à deux. L'instruction WITH permet de choisir les variables qui apparaissent en ligne dans la matrice. Seront croisées uniquement chaque variable de VAR avec chaque variable de WITH.

L'instruction BY sert à distinguer des sous-groupes dans notre population. Les corrélations sont alors calculées pour chacun des sous-groupes, le découpage de la population se faisant suivant les modalités des variables précisées dans l'instruction BY.

L'instruction PARTIAL fournit la liste des variables dont on veut supprimer l'influence dans le calcul des corrélations partielles.

L'instruction WEIGHT permet de définir une variable qui servira à pondérer les observations avant le calcul des corrélations.

Lorsqu'une variable apparaît dans l'instruction FREQ, tout se passe comme si chaque observation i apparaissait v(i) fois dans la table, où v(i) est la valeur en i de la variable désignée par l'instruction FREQ.

Récapitulatif des options de la PROC CORR

par défaut

Ce que la procédure édite	**Comment le supprimer**
Statistiques descriptives univariées	option NOSIMPLE
Matrice des corrélations linéaires	Définir une option KENDALL, SPEARMAN ou HOEFFDING sans préciser PEARSON ou encore : option NOCORR
et des niveaux de significativité de chaque coefficient	option NOPROB

on peut demander en plus

Ce que la procédure peut éditer	**Comment le demander**
tau-b de Kendall	option KENDALL
coefficient de corrélation des rangs de Spearman	option SPEARMAN
mesure de liaison de Hoeffding	option HOEFFDING
coefficient alpha de Cronbach	option ALPHA (avec option PEARSON)
matrice des covariances	option COV (avec option PEARSON)
somme des carrés et des produits	option SSCP (avec option PEARSON)
somme des carrés et des produits des écarts à la moyenne	option CSSCP (avec option PEARSON)
coefficients de corrélation partielle	instruction PARTIAL (sauf avec option HOEFFDING)
variances et écarts-type partiels	instruction PARTIAL (avec option PEARSON)
coefficients de Pearson pondérés	instruction WEIGHT
coefficients calculés sur des sous-populations	instruction BY
matrice des corrélations non carrée	instruction WITH

Lecture de sortie :

The CORR Procedure
3 Variables: salaire subor anciennete

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
salaire	31	1916	1132	59389	968.00000	6097
subor	31	1.96774	5.53464	61.00000	0	30.00000
anciennete	31	7.22581	4.61671	224.00000	0	18.00000

Simple Statistics

Variable	Label
salaire	salaire en euros
subor	nombre de subordonnés
anciennete	

Pearson Correlation Coefficients, N = 31
Prob > |r| under H0: Rho=0

	salaire	subor	anciennete
salaire	1.00000	0.84629	0.34976
salaire en euros		<.0001	0.0538
subor	0.84629	1.00000	0.34600
nombre de subordonnés	<.0001		0.0566
anciennete	0.34976	0.34600	1.00000
	0.0538	0.0566	

Voici l'exemple d'une sortie standard de PROC CORR. Seule l'instruction VAR a été précisée, sans aucune option.
La procédure édite des statistiques univariées sur les trois variables concernées : nombre d'observations non manquantes, moyenne, écart-type, somme, minimum et maximum, ainsi que les labels desdites variables. Ensuite, elle édite la matrice carrée des corrélations linéaires, contenant aussi les niveaux de significativité des coefficients en question.
On conclurait que la rémunération d'un employé semble assez fortement liée au nombre de subordonnés qu'il a ; ces deux variables étant liées, bien que beaucoup plus faiblement, à l'ancienneté de l'employé.

The CORR Procedure
5 Variables: cine musee bibli theatre concert

Spearman Correlation Coefficients, N = 43

cine	cine	theatre	musee	bibli	concert
cine	1.00000	0.21904	-0.13112	-0.12184	-0.00120
musee	musee	bibli	concert	cine	theatre
musee	1.00000	0.45953	-0.16705	-0.13112	0.03526
bibli	bibli	musee	concert	theatre	cine
bibli	1.00000	0.45953	-0.45746	0.15959	-0.12184
theatre	theatre	concert	cine	bibli	musee
theatre	1.00000	-0.24795	0.21904	0.15959	0.03526
concert	concert	bibli	theatre	musee	cine
concert	1.00000	-0.45746	-0.24795	-0.16705	-0.00120

Cette deuxième sortie a été obtenue avec les options SPEARMAN, RANK, NOSIMPLE et NOPROB.
On voit que les gens qui aiment bien aller à la bibliothèque ont tendance à être les mêmes que ceux qui aiment bien aller au musée, alors qu'ils diffèrent assez souvent de ceux qui apprécient les concerts.

Test d'égalité des moyennes avec la PROC TTEST

Nous n'étudions ici que l'aspect de la PROC TTEST qui permet de comparer les moyennes d'une certaine variable pour deux sous-populations.

Dans ce cadre, la procédure PROC TTEST permet de réaliser d'une part un test d'égalité des variances de deux sous-populations, et d'autre part deux tests d'égalité des moyennes (l'un pour le cas où les variances sont égales et l'autre pour le cas où les variances sont significativement différentes).
L'instruction VAR définit la variable sur laquelle on veut faire le test.
L'instruction CLASS induit la variable qui définit les sous-groupes.

La sortie de la PROC TTEST présente d'abord un tableau des moyennes et écart-types pour chacune des deux sous-populations, ainsi que des intervalles de confiance pour chacune de ces quatre grandeurs.
Puis le deuxième tableau présente le résultat du test d'hypothèse nulle : (H0) : moy1 – moy1 = m. Par défaut m=0, et le test est réalisé d'abord dans le cas d'égalité des variances puis dans le cas de variances significativement différentes. On peut définir m $\neq$ 0 grâce à l'option H0.
Le dernier tableau expose le résultat du test d'égalité des variances.

Les tests réalisés par cette procédure nécessitent l'hypothèse de normalité des distributions. On commencera donc toujours par effectuer une PROC UNIVARIATE option NORMAL avec en instruction VAR la variable numérique à laquelle on s'intéresse et en instruction BY la variable qui définit les sous-groupes. Les deux tests de normalité doivent être positifs.

```
PROC TTEST  data=   <options> ;
      CLASS variable ;
      <VAR variable_numérique ;>
      <BY variable ;>
      <WEIGHT variable ;>
```

Les options :

L'option alpha= définit la valeur du seuil pour le calcul des intervalles de confiance.

L'option h0= permet de définir la valeur m telle que l'on teste si l'écart entre les moyennes est égal à m. Par défaut, m=0.

L'option cochran édite l'approximation par la méthode de Cochran en plus de celle par la méthode de Satterthwaite pour le cas où les variances sont significativement différentes.

L'option ci=none supprime l'édition de l'intervalle de confiance pour l'écart-type.

Les instructions :

Comme toujours, l'instruction VAR permet de préciser sur quelles variables (quantitatives) on travaille. Par défaut, le test sera fait pour toutes les variables numériques de la table, sauf celles éventuellement utilisées dans les instructions BY, FREQ et WEIGHT.

L'instruction CLASS définit les sous-groupes sur lesquels on veut effectuer le test. Par exemple, l'instruction `CLASS sexe ;` indique que l'on souhaite comparer les moyennes entre le sous-groupe des hommes et le sous-groupe des femmes.

L'instruction BY sert toujours à définir des sous-populations. Les tests sont alors réalisés pour chacune de ces sous-populations.

L'instruction WEIGHT permet toujours de pondérer les observations.

Lecture d'une sortie standard :

```
proc ttest data=malib.employe;
 var salaire;
 class sexe;
run;
```

The TTEST Procedure
Statistics

Variable	sexe	N	Lower CL Mean	Mean	Upper CL Mean	Lower CL Std Dev	Std Dev	Upper CL Std Dev	Std Err
salaire	F	16	1208.2	1589	1969.8	527.92	714.66	1106.1	178.66
salaire	H	15	1491.7	2264.3	3037	1021.5	1395.3	2200.5	360.26
salaire	Diff (1-2)		-1482	-675.3	131.21	873.87	1097.3	1475.1	394.35

T-Tests

Variable	Method	Variances	DF	t Value	Pr > |t|
salaire	Pooled	Equal	29	-1.71	0.0975
salaire	Satterthwaite	Unequal	20.6	-1.68	0.1082

Equality of Variances

Variable	Method	Num DF	Den DF	F Value	Pr > F
salaire	Folded F	14	15	3.81	0.0146

Il s'agit ici de comparer la rémunération moyenne des hommes à celle des femmes.
On lit que le test d'égalité des variances conclut au rejet de l'hypothèse nulle au seuil de 5%.
Ayant conclu à une différence significative des variances, on s'intéresse au test d'égalité des moyennes selon la méthode de Satterthwaite. Ici si l'on prend un seuil d'erreur de 5%, on accepte l'hypothèse nulle d'égalité des moyennes.
Mais on peut toutefois remarquer que la moyenne féminine : 1589, est bien en-dessous de la moyenne masculine : 2264,3.

En fait le problème ici, c'est qu'on ne peut raisonnablement retenir l'hypothèse de normalité de salaire sur chacun des sous-groupes.

Régression linéaire multiple avec la PROC REG

Cette procédure permet de réaliser la régression linéaire d'une variable numérique *y* sur le sous-espace engendré par les variables numériques $x_1,\ldots,x_p$.

On a : $y=b_0+b_1x_1+\ldots+b_px_p+u$

Où *u* est le résidu et $b_0,\ldots,b_p$ sont les paramètres.

Il est toujours bon de faire quelques graphiques avant toutes choses afin de ne pas se lancer tête baissée dans une régression linéaire qui ne serait pas pertinente (homoscédasticité, forme de la fonction de régression non linéaire…).

```
PROC REG  data=   <options> ;
      MODEL variable_expliquée = variables_explicatives </options> ;
      < BY liste_variables ; >
      < ID liste_variables ; >
      < WEIGHT variable ; >
      < PLOT var_ordonnee*var_abscisse </options> ; >
      < RESTRICT equation ; >
      < <label : > TEST equation ; >
      < OUTPUT out= table_sortie   liste_stats_a_garder ; >
```

Quelques options de la PROC REG :

L'option alpha= définit la valeur du seuil utilisé pour calculer les intervalles de confiance.

L'option corr édite la matrice des corrélations.
L'option simple édite la somme, moyenne, variance, écart-type et somme des carrés pour la variable expliquée et chacune des variables explicatives.
L'option usscp édite les sommes des carrés
L'option all réunit les trois options précédentes.

L'option outest= permet de récupérer dans une table les paramètres estimés. Il est à peu près possible de récupérer tout ce qu'on veut. On se réfèrera à l'aide de SAS pour les autres options du même type (comment garder une trace des coefficients standardisés, de la matrice des covariances, des graphiques,…).

Quelques instructions :

La syntaxe présentée ci-dessus ne comporte pas toutes les instructions qui sont accessibles avec la PROC REG. On se réfèrera à l'aide de SAS pour en savoir plus.

L'instruction MODEL permet de définir la variable à régresser *y* ainsi que la liste des variables explicatives $x_1,\ldots,x_n$.
Elle donne accès à de nombreuses options dont nous parlerons par la suite.

L'instruction BY permet toujours de définir des sous-populations. Dans ce cas, la modélisation est réalisée pour chacune de ces sous-populations.

L'instruction ID définit la variable qui sert d'identifiant à l'observation. En l'absence de cette instruction, c'est le numéro de l'observation qui remplit cet office.

L'instruction WEIGHT permet toujours de pondérer les observations.

L'instruction PLOT permet de tracer des graphes. On peut utiliser les statistiques produites dont la liste est donnée plus bas (auquel cas il faut donner leur mot clé suffixé d'un point) aussi bien que des variables de l'instruction model. On peut aussi utiliser la variable numéro d'observation en l'appelant OBS. D'autres encore sont disponibles, pour les connaître on se reportera éventuellement à l'aide de SAS.
Par exemple, pour tracer le graphe des résidus en fonction de l'observation, on écrit :

```
PLOT r.*obs.;
```

L'instruction RESTRICT permet d'imposer des restrictions sur les paramètres. Par exemple, si l'on sait que deux régresseurs sont liés linéairement par la relation c1*x1=c2*x2, on écrira :

```
RESTRICT c1*x1=c2*x2 ;   ou :   RESTRICT c1*x1-c2*x2 ;
```

Si plusieurs restrictions doivent être imposées, on les sépare par des virgules au sein d'une même instruction RESTRICT. On peut bien sûr aussi utiliser la variable INTERCEPT (qui désigne la constante du modèle).

L'instruction TEST permet de faire des tests sur les paramètres estimés.
Par exemple, si l'on veut tester si $b_1 = b_2$, on écrit : `TEST x1=x2 ;`

L'instruction OUTPUT permet de récupérer certaines des grandeurs calculées dans une table qui est celle indiquée dans OUT= . Il suffit d'en donner la suite sous la forme mot_clé = noms_variables.
Par exemple :

```
OUTPUT OUT=resultat P=yhat R=yres ;
```

stocke dans la table resultat la valeur estimée de y (variable nommée yhat) ainsi que le résidu (variable nommée yres).

Quelques options de l'instruction MODEL :

Par défaut, le modèle intègre une constante parmi les régresseurs. Pour spécifier un modèle sans constante, on peut préciser l'option noint.

Les options que nous évoquerons par la suite sont celles le plus communément utilisées pour étudier :
- L'hétéroscédasticité et l'autocorrélation
- Les individus aberrants
- La colinéarité des régresseurs

et effectuer une sélection de variables.

➢ *L'hétéroscédasticité et l'autocorrélation*

L'option dw permet de réaliser un test de Durbin-Watson. Il s'agit de tester si les résidus sont autocorrélés à l'ordre 1 (c'est-à-dire qu'il existe un réel ρ tel que : $u_n = \rho.u_{n-1}$), l'hypothèse nulle étant qu'il ne l'est pas. Ce test s'applique dans le cas où nos données sont des séries temporelles. On notera que l'option DW ne calcule pas de p-value, il faut donc se référer à la table statistique de Durbin-Watson pour conclure. On rappelle la forme de la statistique de Durbin-Watson :

$$dw = \frac{\sum_{i=2}^{n} (\hat{u}_i - \hat{u}_{i-1})^2}{\sum_{i=1}^{n} \hat{u}_i^{\,2}}$$

La table statistique fournit le donnée de deux seuils d_L et d_U. La région critique coïncide alors avec $\{dw < d_L\} \cup \{dw > d_U\}$

L'option spec (de la PROC REG et non de l'instruction MODEL !) réalise un test de White pour diagnostiquer l'hétéroscédasticité éventuelle. L'hypothèse nulle est que la variance des résidus est une constante.

➢ *Les individus aberrants*

Les options r et influence permettent d'éditer un certain nombre de grandeurs servant à repérer les individus atypiques et les individus influents. Lorsqu'un individu se distinguera particulièrement du lot, c'est-à-dire lorsqu'il se trouvera un individu pour être à la fois atypique et influent, et ce de manière ostentatoire, on envisagera de supprimer cet individu de notre étude.

Examinons les règles empiriques relatives à la détection des observations :

➢ **Atypiques**

- Pour les variables explicatives :

On calcule alors le levier h_{ii} qui est le i^{ème} élément diagonal de la Hat matrice *H*

$$h_{ii} = x_i'(X'X)^{-1}x_i$$
$$H = X(X'X)^{-1}X'$$

Si $h_{ii} > 2\frac{p+1}{n}$, alors l'observation i est considérée comme atypique au regard des régresseurs. C'est d'autant plus vrai que h_{ii} est proche de 1.

- Pour la variable expliquée :

On calcule les résidus studentisés r_i :

$$r_i = \frac{\hat{u}_i}{\sqrt{\hat{V}(\hat{u}_i)}} = \frac{Y_i - \hat{Y}_i}{\hat{\sigma}\sqrt{1-h_{ii}}}$$

et les résidus studentisés à validation croisée R_i : c'est le résidu studentisé en i calculé sur l'estimation du modèle dans lequel on a ôté i. R_i mesure l'influence qu'a l'observation i sur sa propre estimation.

Si la valeur absolue de ces grandeurs est très supérieure à 2, alors l'observation est considérée comme atypique au regard de la variable expliquée.

➢ **Influentes** :

Note : l'indice –i sera aposé pour préciser que la grandeur est calculée pour le modèle privé de l'observation i.
Note : on désignera le vecteur des paramètres indifféremment par b ou par □□

- De manière globale :

On peut calculer l'agrégat dit PRESS (Predicted Residuals Sum of Squares) :

$$PRESS = \sum_{i=1}^{n}\left(Y_i - \hat{Y}_{-i,i}\right)^2$$

Plus le PRESS s'éloigne de la somme des carrés des résidus (SSE), plus on peut suspecter l'existence d'observations influentes.

- Pour l'estimation des paramètres *b* :

On calcule la distance de Cook qui mesure en quelque sorte la distance entre le vecteur des paramètres estimés *b* et ce même vecteur calculé sur le modèle privé de l'observation i :

$$D_i = \frac{(\hat{\beta} - \hat{\beta}_{-i})'X'X(\hat{\beta} - \hat{\beta}_{-i})}{\hat{\sigma}^2(p+1)}$$

lorsque $D_i > 1$ on conclura à l'influence de l'observation.

- Pour l'estimation du paramètre b_j :

On calcule la quantité :

$$DFBETAS_{ij} = \frac{\hat{\beta}_j - \hat{\beta}_{-i,j}}{\hat{\sigma}_{-i}\sqrt{(X'X)^{-1}_{jj}}}$$

Il s'agit de comparer les estimations de b_j selon si i appartient ou non au modèle. Si $\left|DFBETAS_{ij}\right| > \frac{2}{\sqrt{n}}$, alors on considère que l'observation i est trop influente dans l'estimation du paramètre b_j.

- Pour l'estimation de Y en i :

On calcule une quantité qui mesure l'influence de i sur sa propre valeur ajustée :

$$DFFITS_i = \frac{\hat{Y}_i - \hat{Y}_{-i,i}}{\hat{\sigma}_{-i}\sqrt{h_{ii}}}$$

Lorsque $\left|DFFITS_i\right| > 2\sqrt{\frac{p+1}{n}}$, l'observation semble trop influente.

- Pour la précision de l'estimation du vecteur des paramètres :

On calcule le covratio :

$$CovRatio_i = \frac{\left(\hat{\sigma}^2{}_{-i}\right)^{p+1}\det(X'X)}{\left(\hat{\sigma}^2\right)^{p+1}\det(X'_{-i}X_{-i})}$$

Si $\left|CovRatio_i - 1\right| > 3\frac{p+1}{n}$, l'observation semble louche, et ce d'autant plus que $CovRatio_i$ est éloigné de 1.

Récapitulatif : Quelle option fait quoi ?

	R	**INFLUENCE**
atypique	résidus studentisés (Student Residuals)	résidus Studentisés à validation croisée (Rstudent)
		leviers (Hat Diag)
influente	PRESS	
	distance de Cook (Cook's D)	DFFITS
		DFBETAS
		CovRatio

➢ *La colinéarité des régresseurs*

L'existence de colinéarité entre les régresseurs sous-tend une certaine redondance d'information dans le modèle, les conséquences néfastes se situant à la fois sur le plan statistique et sur celui de l'interprétation des coefficients.

Mentionnons d'abord l'option i qui permet d'éditer la matrice $(X'X)^{-1}$.

Mentionnons également que l'on peut se faire une première idée de l'existence ou non de colinéarité en comparant la sortie de l'option ss1 avec celle de l'option ss2.

L'option SS1 calcule pas à pas pour chacun des régresseurs (avec l'ordre donné dans l'instruction MODEL) la variation de somme des carrés estimés due à l'introduction dudit régresseur.
L'option SS2 calcule la même variation, toujours pour chacun des régresseurs, mais en supposant à chaque fois que le régresseur en question est le dernier à être introduit dans le modèle.
Si les deux sorties sont très différentes, on se doute qu'il existe un problème de colinéarité.

De manière plus précise, on utilise deux types d'options :

➢ Les options vif et tol calculent respectivement le facteur d'influence de la variance et la tolérance. Comme TOL est l'inverse de VIF, on n'éditera qu'une des deux grandeurs.

On a $TOL_j = 1 - R_j{}^2$, où R_j^2 est le coefficient de détermination (le R²) de la régression de X_j sur les autres régresseurs. Dans le cas extrême où X_j est combinaison linéaire des autres régresseurs, R_j^2

= 1. On en déduit que si TOL_j < 0,1 (ou VIF_j > 10), on peut suspecter une situation de colinéarité pour le régresseur X_j, et ce d'autant plus que TOL_j est proche de 0 (que VIF_j est grand).

➢ Les options collin et collinoint permettent de préciser le nombre de relations de colinéarité existant. Ces deux options calculent les indices de conditionnement CI_j ainsi que les proportions de variance notées $VarProp_{jk}$:

$$CI_j = \sqrt{\frac{\lambda_{.1}}{\lambda_{.j}}}$$

$$Var\,\mathrm{Pr}op_{jk} = \frac{j^{ème}composante\,de Var(\hat{b}_k)}{Var(\hat{b}_k)}$$

où les λ_j sont les valeurs propres de la matrice des corrélations des régresseurs, ordonnées de manière décroissante.
notant que la variance du k^{ème} coefficient estimé se décompose en la somme de p composantes de type α_{kj} / λ_j .

COLLIN s'utilise lorsque le modèle spécifie parmi ses régresseurs une constante (qui a une signification physique). Dans les autres cas, on utilise COLLINOINT.

On considère qu'un CI_j > 30 est louche. L'existence d'une relation de colinéarité apparaît d'autant plus évidente que CI_j est grand. Pour savoir quels régresseurs elle met en cause, on regarde ensuite les $VarProp_{jk}$ pour ce j, et ceux qui sont supérieurs à 0,5 désignent les coupables.

➢ *La sélection de variables*

L'option selection= réalise des procédures de sélection des régresseurs les plus pertinents parmi ceux proposés dans l'instruction MODEL. Les critères de choix des variables varient selon la méthode choisie. Suivent les méthodes les plus utilisées :

➢ L'option selection=rsquare fournit la liste des modèles possibles par ordre croissant de nombre de variables explicatives puis par ordre décroissant de R².

➢ L'option selection=cp utilise le critère du Cp de Mallows pour classer les sous-modèles possibles. Cette statistique fournit une mesure de l'erreur quadratique moyenne. Lorsqu'un sous-modèle à k régresseurs est proche du modèle complet au sens de l'EQM, on doit avoir Cp ≈ k+1.

➢ L'option selection=forward procède à une sélection des variables en ce sens qu'elle propose un sous-modèle. La procédure consiste à intégrer pas à pas, en partant du modèle avec la seule constante, le régresseur qui induit le plus important gain de somme des carrés expliqués. La procédure s'arrête lorsque le gain en question devient inférieur à un certain seuil que l'on peut modifier.

➢ L'option selection=backward repose sur le même principe que la procédure forward, sauf que cette fois on part du modèle complet et on retire pas à pas le régresseur qui induit la plus petite augmentation de somme des carrés résiduels.

➢ L'option selection=stepwise, enfin, est du même type que forward, sauf qu'il est possible que des variables introduites à une certaine étape soient retirées du sous-modèle dans une autre étape. Elle combine donc procédure forward et procédure backward.

Liste des grandeurs que l'on peut récupérer dans PLOT ou dans OUTPUT (et mots-clés associés) :

Cookd	statistique D de Cook relative à l'influence d'une observation
Covratio	mesure de l'influence de l'observation sur la covariance des paramètres
Dffits	mesure de l'influence de l'observation sur la valeur estimée de *y*
H	levier de l'observation

Lcl (resp. Ucl) borne inférieure (resp. supérieure) de l'intervalle de confiance pour la valeur prédite y_i
Lclm (resp. Uclm) borne inférieure (resp. supérieure) de l'intervalle de confiance pour la moyenne de *y*

Predicted ou P	estimation de la variable expliquée
Residual ou R	résidu de la régression = $y - y_{predicted}$
Student	résidus studentisés
Rstudent	résidus studentisés à validation croisée
Press	résidu rapporté à 1-H

Stdi	écart-type de l'estimation de y_i
Stdp	écart-type de l'estimation de la moyenne de *y*
Stdr	écart-type des résidus estimés

Lecture de sorties :

- *Une sortie standard :*

```
proc reg data=malib.employe;
 model salaire = anciennete subor age;
run;
```

The REG Procedure
Model: MODEL1
Dependent Variable: salaire salaire en euros

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	27848720	9282907	23.65	<.0001
Error	27	10597854	392513		
Corrected Total	30	38446573			

Root MSE	626.50866	R-Square	0.7243
Dependent Mean	1915.77419	Adj R-Sq	0.6937
Coeff Var	32.70264		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1117.36274	573.88399	1.95	0.0620
anciennete	1	-9.21669	46.23709	-0.20	0.8435
subor	1	167.11617	22.13012	7.55	<.0001
age	1	14.21831	21.51873	0.66	0.5144

La sortie standard contient donc :

Un bloc d'analyse de la variance (décomposition de la somme des carrés totaux en somme des carrés résiduels (ligne Error) et somme des carrés estimés (ligne Model)).
A ce niveau est aussi réalisé un test de Fischer de validité globale du modèle. L'hypothèse nulle est (H0) : les paramètres sont tous simultanément nuls. Ici la p-value est très inférieure à 0,5, donc on rejette (H0), ce qui signifie que le modèle est globalement valide.

La colonne DF indique le nombre de degrés de liberté (degrees of freedom). Le DF de la ligne Model représente p, c'est à dire le nombre de régresseurs, constante exclus. Le DF de la ligne Corrected Total vaut n-1 où n est la taille de la population. La colonne Mean Square calcule le rapport de Sum of Squares au DF.

Un bloc intermédiaire : Root MSE est l'écart-type estimé. Dependent Mean est la moyenne de y. Coeff Var est le coefficient de variation, défini comme le rapport des deux grandeurs précédentes multiplié par 100. Enfin on a la donnée du R^2 (rapport de la somme des carrés estimés à la somme des carrés totaux) et du R^2 ajusté. Notons que le R^2 n'est pas interprétable dans un modèle sans constante.

Un bloc d'estimation des paramètres liste les régresseurs, avec une estimation du paramètre associé, l'écart-type estimé pour ledit paramètre, ainsi que la statistique et la p-value relatives à un test de Student de nullité du paramètre (l'hypothèse nulle étant justement la nullité du paramètre). Ici on voit que les variables anciennete et age ne sont pas significatives.

- *On remanie le programme en y incluant plusieurs options comme suit :*

```
proc reg data=malib.employe corr;
 model salaire = anciennete subor age / dw spec vif collinoint r
influence ;
 test anciennete=age ;
 plot p.*obs. ;
run;
```

The REG Procedure

Correlation

Variable	Label	anciennete	subor	age	salaire
anciennete		1.0000	0.3460	0.8419	0.3498
subor	nb subordonnés	0.3460	1.0000	0.3400	0.8463
age		0.8419	0.3400	1.0000	0.3704
salaire	salaire en E	0.3498	0.8463	0.3704	1.0000

Dependent Variable: salaire salaire en E

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	27848720	9282907	23.65	<.0001
Error	27	10597854	392513		
Corrected Total	30	38446573			

Root MSE	626.50866	R-Square	0.7243
Dependent Mean	1915.77419	Adj R-Sq	0.6937
Coeff Var	32.70264		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation
Intercept	1	1117.36274	573.88399	1.95	0.0620	0
anciennete	1	-9.21669	46.23709	-0.20	0.8435	3.48267
subor	1	167.11617	22.13012	7.55	<.0001	1.14660
age	1	14.21831	21.51873	0.66	0.5144	3.46648

Collinearity Diagnostics(intercept adjusted)

Number	Eigenvalue	Condition Index	Proportion of Variation: anciennete	subor	age
1	2.06323	1.00000	0.05771	0.07283	0.05775
2	0.77872	1.62773	0.03082	0.92697	0.03287
3	0.15805	3.61301	0.91147	0.00020110	0.90938

Test of First and Second Moment Specification

DF	Chi-Square	Pr > ChiSq
9	12.49	0.1873

Durbin-Watson D	0.852
Number of Observations	31
1st Order Autocorrelation	0.484

Output Statistics

Obs	Dep Var salaire	Predicted Value	Std Error Mean Predict	Residual	Std Error Residual	Student Residual	-2-1 0 1 2	Cook's D
1	3963	2653	171.6793	1310	602.5	2.175	|**** |	0.096
2	2088	1550	128.3430	537.5132	613.2	0.877	|* |	0.008
3	1905	1903	213.4522	1.7724	589.0	0.00301	| |	0.000
4	1875	1575	146.2143	300.2916	609.2	0.493	| |	0.003
5	1356	1535	138.7658	-179.4820	610.9	-0.294	| |	0.001
6	1394	1641	141.6958	-246.7983	610.3	-0.404	| |	0.002
7	1379	1455	190.6224	-76.1737	596.8	-0.128	| |	0.000
8	1844	2280	257.8219	-436.1149	571.0	-0.764	*| |	0.030
9	1097	1435	205.9660	-338.1672	591.7	-0.572	*| |	0.010
10	1372	1627	167.6276	-255.3666	603.7	-0.423	| |	0.003
11	1417	1790	326.9184	-373.3427	534.5	-0.699	*| |	0.046
12	1379	1689	190.2358	-310.2414	596.9	-0.520	*| |	0.007
13	1033	1526	131.9155	-493.2653	612.5	-0.805	*| |	0.008
14	1234	2102	387.5017	-867.7765	492.3	-1.763	***| |	0.481
15	1021	1440	194.9316	-419.1689	595.4	-0.704	*| |	0.013
16	1067	1626	172.5952	-558.7935	602.3	-0.928	*| |	0.018
17	6097	6775	590.4819	-677.6047	209.4	-3.236	|****** |	20.820
18	4268	2626	207.2753	1642	591.2	2.778	| |*****	0.237
19	3353	2533	199.7933	819.8226	593.8	1.381	| |**	0.054
20	2286	1693	176.9587	593.4051	601.0	0.987	| |*	0.021
21	1905	1679	138.0213	225.8369	611.1	0.370	| |	0.002
22	2507	1751	119.4985	755.9587	615.0	1.229	| |**	0.014
23	2210	1654	217.4930	555.8449	587.5	0.946	| |*	0.031
24	1951	1585	188.4737	366.2884	597.5	0.613	| |*	0.009
25	2241	1900	244.1068	341.2009	577.0	0.591	| |*	0.016
26	1347	1687	299.4044	-339.8818	550.3	-0.618	*| |	0.028
27	1353	1484	161.1971	-130.6104	605.4	-0.216	| |	0.001
28	1361	1607	126.6470	-246.3601	613.6	-0.402	| |	0.002
29	1036	1574	121.7014	-537.9219	614.6	-0.875	*| |	0.008
30	1082	1612	158.8746	-529.5752	606.0	-0.874	*| |	0.013
31	968	1402	236.1441	-433.7290	580.3	-0.747	*| |	0.023

Output Statistics

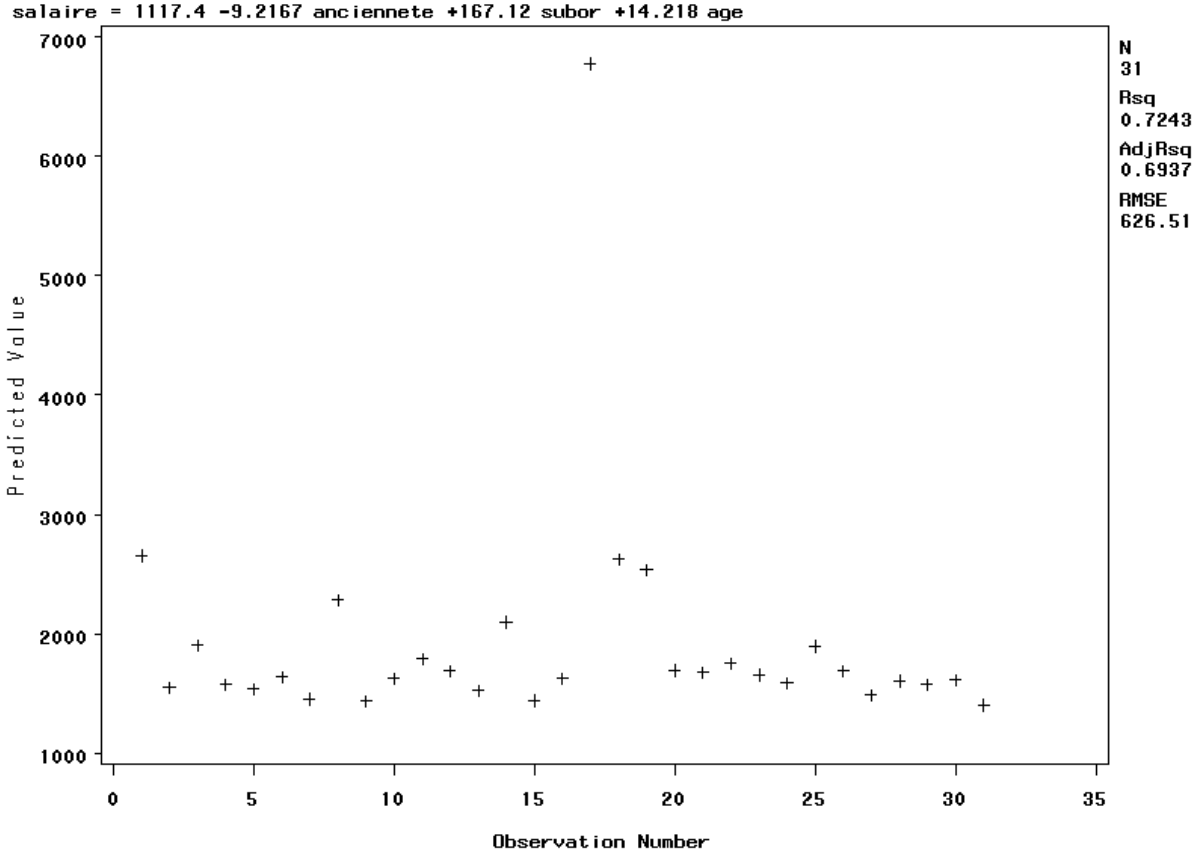
Obs	RStudent	Hat Diag H	Cov Ratio	DFFITS	DFBETAS Intercept	DFBETAS anciennete	DFBETAS subor	DFBETAS age
1	2.3497	0.0751	0.5820	0.6695	-0.2275	-0.3772	0.3181	0.3600
2	0.8727	0.0420	1.0815	0.1826	0.0918	0.0606	-0.0522	-0.0648
3	0.002953	0.1161	1.3157	0.0011	-0.0008	-0.0004	-0.0003	0.0008
4	0.4859	0.0545	1.1863	0.1166	0.0469	0.0654	-0.0429	-0.0454
5	-0.2887	0.0491	1.2074	-0.0656	-0.0203	0.0204	0.0067	-0.0018
6	-0.3981	0.0512	1.1962	-0.0924	0.0315	0.0227	0.0369	-0.0438
7	-0.1253	0.0926	1.2785	-0.0400	-0.0357	-0.0192	0.0019	0.0298
8	-0.7577	0.1694	1.2829	-0.3421	0.2724	0.1918	0.0317	-0.2940
9	-0.5643	0.1081	1.2419	-0.1964	-0.1348	0.0218	-0.0198	0.0725
10	-0.4165	0.0716	1.2197	-0.1157	-0.0091	-0.0595	0.0518	0.0174
11	-0.6918	0.2723	1.4857	-0.4232	0.1507	-0.1647	0.1893	-0.0691
12	-0.5126	0.0922	1.2306	-0.1634	0.0511	-0.0379	0.0780	-0.0381
13	-0.8000	0.0443	1.1041	-0.1723	-0.0916	-0.0040	0.0256	0.0447
14	-1.8388	0.3826	1.1550	-1.4474	-1.1639	-1.3134	-0.1305	1.3126
15	-0.6973	0.0968	1.1956	-0.2283	-0.1739	-0.0066	-0.0156	0.1083
16	-0.9253	0.0759	1.1055	-0.2652	0.1072	0.1853	0.0524	-0.1789
17	-4.0588	0.8883	1.4618	-11.4459	0.6305	0.3682	-10.5110	-0.3359
18	3.2250	0.1095	0.3399	1.1306	0.3527	0.7929	0.2443	-0.5039
19	1.4054	0.1017	0.9659	0.4729	0.1251	-0.2068	0.2985	0.0176
20	0.9869	0.0798	1.0909	0.2906	0.0517	-0.1690	0.0378	0.0567
21	0.3636	0.0485	1.1977	0.0821	0.0553	0.0108	0.0027	-0.0344
22	1.2414	0.0364	0.9586	0.2412	0.0697	0.0670	-0.0530	-0.0425
23	0.9441	0.1205	1.1555	0.3495	0.0917	-0.2004	0.0677	0.0391
24	0.6058	0.0905	1.2089	0.1911	0.0742	0.1411	-0.0698	-0.0910
25	0.5841	0.1518	1.3014	0.2471	-0.0761	0.0930	-0.0966	0.0365
26	-0.6104	0.2284	1.4238	-0.3321	0.2042	0.2951	0.0326	-0.2849
27	-0.2119	0.0662	1.2368	-0.0564	-0.0455	-0.0196	0.0047	0.0346
28	-0.3952	0.0409	1.1838	-0.0816	0.0100	0.0181	0.0290	-0.0261
29	-0.8714	0.0377	1.0771	-0.1726	-0.0246	0.0303	0.0473	-0.0197
30	-0.8699	0.0643	1.1081	-0.2280	0.0715	0.1477	0.0460	-0.1362
31	-0.7412	0.1421	1.2468	-0.3016	-0.2224	0.0206	-0.0414	0.1311

Sum of Residuals	0
Sum of Squared Residuals	10597854
Predicted Residual SS (PRESS)	50241922

The REG Procedure
Model: MODEL1

Test 1 Results for Dependent Variable salaire

Source	DF	Mean Square	F Value	Pr > F
Numerator	1	50909	0.13	0.7215
Denominator	27	392513		



L'option CORR édite la matrice des corrélations. On peut déjà repérer que les variables age et anciennete sont fortement corrélées.

Le graphique des valeurs prédites n'invalide pas notre choix de modèle : la forme fonctionnelle n'est pas mauvaise a priori, et on ne repère pas d'hétéroscédasticité au premier coup d'œil. Par contre, on repère une valeur aberrante.

L'option DW réalise un test de Durbin Watson. Dans une table statistique de Durbin Watson, on pourrait relever les seuils relatifs à notre modèle, et on déduirait que l'hypothèse d'autocorrélation à l'ordre 1 peut être acceptée à 5%. Cela dit, ici cela n'a pas beaucoup de sens. Dans un pareil cas, on regarderait plutôt le test de White réalisé par l'option SPEC. Ici l'hypothèse nulle d'homoscédasticité est acceptée à 5%.

L'analyse des observations atypiques et influentes par les options R et INFLUENCE montre que :
Les observations 1, 17 et 18 sont atypiques pour le salaire. Dans la table on peut vérifier qu'il s'agit des trois plus gros salaires, et que les 17 et 18 sont particulièrement éloignés du reste des valeurs.
L'observation 17, et à une moindre mesure l'observation 14, sont atypiques pour les régresseurs. Effectivement, on remarque que la 17 a 30 subordonnés (les autres valeurs se situant entre 0 et 6) et un âge et une ancienneté assez élevés (55 ans et 15 ans) par rapport à la moyenne. Quant à l'observation 14, dont le caractère atypique est moins marqué, on peut noter qu'elle présente une ancienneté et un nombre de subordonnés un peu plus élevé que la moyenne alors que son âge est plutôt inférieur à la moyenne.
L'observation 17 se démarque nettement du lot en ce qui concerne son influence sur l'estimation. La distance de Cook est fortement élevée pour cette observation. La 17 influe sur sa propre prédiction, et particulièrement sur l'estimation du coefficient de la variable subor, ce qui semble logique.
En conclusion de cette analyse, on décidera de supprimer de notre régression l'observation 17.

A ce stade il n'est pas nécessaire de poursuivre l'analyse. On recommencerait la procédure après avoir ôté l'observation aberrante. Mais puisqu'il ne s'agit ici que de savoir lire des sorties, continuons :

Grâce à l'option VIF, on repère que ces variables ont justement un facteur d'influence de la variance plus élevé que les autres ; mais ceci n'est pas flagrant. On porte donc notre regard sur la sortie de l'option COLLINOINT. Même si les CI ne sont pas très importants, les Proportion of Variation sont tels que l'on peut au moins soupçonner une relation de colinéarité entre l'âge et l'ancienneté.
A ce stade, on peut alors décider de supprimer une des deux variables impliquées dans la relation de colinéarité.

➢ *Que proposent les procédures de sélection ?*

Supprimons l'observation 17 et réitérons la PROC REG, d'abord avec l'option SELECTION=CP puis avec l'option SELECTION=RSQUARE, enfin avec l'option SELECTION=STEPWISE.

The REG Procedure
Model: MODEL1
Dependent Variable: salaire

C(p) Selection Method

Number in Model	C(p)	R-Square	Variables in Model
1	1.7009	0.6609	subor
2	2.3790	0.6771	age subor
2	3.3471	0.6652	anciennete subor
3	4.0000	0.6817	age anciennete subor
1	51.9024	0.0463	age
1	52.5471	0.0384	anciennete
2	53.8140	0.0474	age anciennete

La procédure n'apporte pas grand chose. Elle semble désigner le modèle que l'on a choisi, mais les trois précédents ne sont pas fondamentalement plus mauvais.

The REG Procedure
Model: MODEL1
Dependent Variable: salaire

R-Square Selection Method

Number in Model	R-Square	Variables in Model
1	0.6609	subor
1	0.0463	age
1	0.0384	anciennete
2	0.6771	age subor
2	0.6652	anciennete subor
2	0.0474	age anciennete
3	0.6817	age anciennete subor

Il s'agit d'arbitrer entre biais et précision. Le premier souci commande de ne pas trop choisir de régresseurs, tandis que le deuxième pousse à améliorer le R^2. Ici on voit que le meilleur R^2 pour les modèles à 1 variable n'est guère inférieur au meilleur des R^2 (qui correspond au modèle total). On est donc amené à choisir le modèle avec le seul régresseur subor.

La procédure stepwise conduit-elle au même résultat ?

```
                                The REG Procedure
                                  Model: MODEL1
                    Dependent Variable: salaire salaire en euros

                              Stepwise Selection: Step 1
                Variable subor Entered: R-Square = 0.6609 and C(p) = 1.7009

                                   Analysis of Variance

                                            Sum of           Mean
        Source                    DF        Squares         Square    F Value    Pr > F
        Model                      1       13469424       13469424      54.57    <.0001
        Error                     28        6911745         246848
        Corrected Total           29       20381169

                             Parameter     Standard
              Variable        Estimate        Error   Type II SS  F Value  Pr > F
              Intercept     1409.71705    103.40401     45879515   185.86  <.0001
              subor          354.85447     48.03863     13469424    54.57  <.0001

                                Bounds on condition number: 1, 1
--------------------------------------------------------------------------------
                              Stepwise Selection: Step 2
                Variable age Entered: R-Square = 0.6771 and C(p) = 2.3790

                                   Analysis of Variance
                                            Sum of           Mean
        Source                    DF        Squares         Square    F Value    Pr > F

        Model                      2       13799241        6899620      28.30    <.0001
        Error                     27        6581928         243775
        Corrected Total           29       20381169

                             Parameter     Standard
              Variable        Estimate        Error   Type II SS  F Value  Pr > F
              Intercept      997.63890    368.87468      1783111     7.31  0.0117
              subor          348.75934     48.02542     12855788    52.74  <.0001
              age             11.26687      9.68638       329817     1.35  0.2549

                            Bounds on condition number: 1.012, 4.0482
--------------------------------------------------------------------------------
                              Stepwise Selection: Step 3
                Variable age Removed: R-Square = 0.6609 and C(p) = 1.7009

                                   Analysis of Variance
                                            Sum of           Mean
        Source                    DF        Squares         Square    F Value    Pr > F

        Model                      1       13469424       13469424      54.57    <.0001
        Error                     28        6911745         246848
        Corrected Total           29       20381169

                             Parameter     Standard
              Variable        Estimate        Error   Type II SS  F Value  Pr > F
              Intercept     1409.71705    103.40401     45879515   185.86  <.0001
              subor          354.85447     48.03863     13469424    54.57  <.0001

                                Bounds on condition number: 1, 1
--------------------------------------------------------------------------------

             All variables left in the model are significant at the 0.1500 level.
    The stepwise method terminated because the next variable to be entered was just
removed.
```

Summary of Stepwise Selection

Step	Variable Entered	Variable Removed	Number Vars In	Partial R-Square	Model R-Square	C(p)	F Value	Pr > F
1	subor		1	0.6609	0.6609	1.7009	54.57	<.0001
2	age		2	0.0162	0.6771	2.3790	1.35	0.2549
3		age	1	0.0162	0.6609	1.7009	1.35	0.254

La procédure s'arrête au bout de la troisième étape. Le modèle final proposé est bien celui constitué du seul régresseur subor.
Notons que la procédure forward proposerait de garder aussi la variable age.

A ce stade on recommencerait la régression soit avec subor seulement, soit avec subor et age.
L'estimation du deuxième modèle indique que la variable age n'apparaît pas significative. Le modèle que l'on retient est donc : salaire = 1410 + 355 * subor

Régression sur variables catégorielles avec la PROC LOGISTIC

Lorsque l'on souhaite effectuer une régression sur une variable qualitative, la régression linéaire classique (voir paragraphe précédent) n'est plus adaptée. Il faut mettre en œuvre d'autres méthodes. L'une d'elle est la régression logistique.

Considérons y une variable dichotomique, y ∈ {0 ; 1}, et x1 x2 x3 des régresseurs (variables quantitatives continues). Le modèle logistique correspondant à l'instruction suivante :

```
MODEL y = x1 x2 x3 ;
```

Correspond au modèle suivant :

$$Log\left(\frac{P[y=0]}{P[y=1]}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

Par défaut, SAS modélise la probabilité de l'événement correspondant à la plus basse des modalités de y. La méthode utilisée est la maximisation de la vraisemblance de y sachant x.

Les tests réalisés relativement à la nullité des coefficients sont des tests de Wald. Concernant la validité globale du modèle, le test du score et le test du rapport des maxima de vraisemblance sont édités (on rappelle qu'ils sont équivalents asymptotiquement), en plus des critères d'Aikaike et de Schwartz. On rappelle que parmi une liste de modèles emboîtés, un modèle est d'autant meilleur que ces critères sont faibles.

La PROC LOGISTIC a cet avantage sur la PROC CATMOD (autre procédure qui permet de réaliser des régressions sur variables catégorielles) qu'elle édite les odd-ratios.
Si l'odd-ratio de la variable x_i vaut w, cela s'interprète comme suit : « Lorsque x_i augmente d'une unité, les chances $\frac{y=a}{y\neq a}$ sont multipliées par w »

Enfin la procédure calcule le nombre de paires concordantes (dont la prévision concorde avec la vraie valeur) ainsi que quatre statistiques obtenues à partir du nombre de paires concordantes, discordantes et tied : D de Sommer, γ, τ_a et c. Plus ces valeurs sont grandes, et meilleure est l'adéquation du modèle aux données.

```
PROC LOGISTIC  data=   <options> ;
      < CLASS var_quali ; >
      MODEL y = variables </options> ;
      < CONTRAST 'label' l11 l12 … l1n … lm1 … lmn </options> ; >
      < <label : > TEST equation ; >
      < BY listevariables ; >
      < FREQ variable ; >
      < WEIGHT listevariables ; >
      < OUTPUT out= table_sortie   liste_stats_a_garder ; >
```

Les instructions :

La syntaxe présentée ci-dessus ne comporte pas toutes les instructions qui sont accessibles avec la PROC LOGISTIC. On se réfèrera à l'aide de SAS pour en savoir plus.

Une instruction MODEL doit être spécifiée. Elle sera précédée d'une instruction CLASS si le modèle comporte des variables qualitatives.

Comme dans la PROC REG, l'instruction TEST permet de faire des tests sur les paramètres estimés. Par exemple, si l'on veut tester si $\beta_1 = \beta_2$, on écrit : `TEST x1=x2 ;`

L'instruction CONTRAST permet également d'effectuer des tests sur les paramètres estimés. Plus exactement, elle permet de tester (H0) : $L\beta = 0$ où L est la matrice dont l_{ij} est l'élément à la ligne i, colonne j ; n est le nombre de variables du modèle et m le nombre d'équations du système testé.

L'instruction BY permet toujours de définir des sous-populations. Dans ce cas, la modélisation est réalisée pour chacune de ces sous-populations.

L'instruction FREQ permet de définir une variable entière dont les valeurs indique le nombre de fois que l'observation courante doit être comptée.

L'instruction WEIGHT permet toujours de pondérer les observations.

L'instruction OUTPUT permet de récupérer certaines des grandeurs calculées dans une table qui est celle indiquée dans OUT= . Il suffit d'en donner la suite sous la forme mot_clé = noms_variables.
Par exemple :

```
OUTPUT OUT=resultat P=yhat H=hatdiag1 hatdiag2 hatdiag3 ;
```

stocke dans la table resultat la valeur estimée de y (variable nommée yhat) ainsi que la valeur du levier pour chacun des régresseurs (variables nommée yres) – voir paragraphe sur les observations atypiques dans la description de la PROC REG.

Les options :

L'option simple permet d'éditer des statistiques descriptives simples sur les variables du modèle.

L'option outest= permet de spécifier le nom d'une table dans laquelle on récupère les paramètres estimés (ainsi que leurs covariances estimées dans le cas où l'on ajoute l'option covout).

L'option descending permet de considérer la probabilité de réalisation de la modalité la plus élevée (par défaut c'est la plus basse).

L'option order= permet de choisir l'ordre des modalités de *y* dans le traitement. order=data ordonne selon l'ordre dans lequel les modalités apparaissent dans la table. order=freq ordonne de la modalité la plus fréquente à la plus rare.

➢ *Quelques options de l'instruction MODEL :*

L'option link= permet de choisir une autre fonction de lien que la logit : probit, normit ou cloglog.

L'option noint supprime la constante du modèle.

Comme dans la PROC REG, l'option influence sert à repérer les observations influentes. On peut la coupler avec l'option iplots qui génère des graphiques.

Comme dans la PROC REG, on peut utiliser l'option selection= .

Lecture de sorties standards :

```
proc logistic data=malib.employe descending;
 model prime= anciennete subor;
run;
```

The LOGISTIC Procedure

Model Information

Data Set	MALIB.EMPLOYE
Response Variable	prime
Number of Response Levels	2
Number of Observations	115
Link Function	Logit
Optimization Technique	Fisher's scoring

Response Profile

Ordered Value	prime	Total Frequency
1	O	24
2	N	91

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	119.811	89.664
SC	122.556	97.898
-2 Log L	117.811	83.664

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	34.1477	2	<.0001
Score	27.9854	2	<.0001
Wald	19.4717	2	<.0001

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-3.3874	0.6221	29.6486	<.0001
anciennete	1	0.1828	0.0613	8.8792	0.0029
subor	1	0.1444	0.0487	8.7776	0.0030

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
anciennete	1.201	1.065	1.354
subor	1.155	1.050	1.271

Association of Predicted Probabilities and Observed Responses

Percent Concordant	90.0	Somers' D	0.802
Percent Discordant	9.8	Gamma	0.804
Percent Tied	0.3	Tau-a	0.267
Pairs	2184	c	0.901

La sortie standard est composée de 7 tableaux :

Le premier comporte des informations sur le modèle : variable régressée, son nombre de modalités, la fonction de lien… Ici on régresse la variable prime de la table malib.employe, qui a deux modalités.

Le deuxième présente les modalités de la variable régressée et la fréquence de chacune d'elles. 24 employés ont eu une prime cette année et 91 n'en ont pas eu. C'est la probabilité d'avoir une prime qui est analysée par le modèle, ceci grâce à l'option descending.
Le modèle estimé est :

$$\ln\left(\frac{\mathrm{P}(prime = oui)}{\mathrm{P}(prime = non)}\right) = b_0 + b_1 * anciennete + b_2 * subor$$

Un message informe de ce que le modèle est convergent.

Le troisième tableau présente les critères d'Aikaike, de Schwartz et du rapport des maxima de vraisemblance calculés pour le modèle avec la seule constante comme régresseur et pour le modèle spécifié dans la procédure. Les valeurs de ces critères étant inférieures pour notre modèle que dans le cas de la constante seule, notre modèle n'est a priori pas trop mauvais.

Le quatrième tableau résume les trois tests effectués pour la validité globale du modèle. Ici le modèle est globalement valide puisqu'on rejette l'hypothèse de nullité simultanée des paramètres.

Le cinquième tableau nous donne enfin le loisir de lire les paramètres estimés et leur significativité. Ici tous les coefficients sont significativement différents de zéro. Il est à noter qu'on n'utilise pas directement la valeur de ces coefficients. La manière dont on les interprète dépend uniquement de leur signe et de leurs ordres de grandeur comparés. Ici on conclut que plus on a d'ancienneté et plus on a de chances d'avoir une prime. De même plus on a de subordonnés, plus on a de chances d'avoir une prime. Les effets de ces deux variables étant comparables.

Le sixième tableau est celui des odds ratio. Dans le cas qui nous intéresse ici l'interprétation en est simple : lorsqu'on gagne un an d'ancienneté, le rapport des chances d'avoir une prime sur les risques de ne pas avoir de prime est multiplié par 1,201.

Le dernier tableau permet de constater si le modèle est bien adapté aux données ou non. Ici les prévisions coïncident à 90% avec la réalité observée, donc le modèle est bien adapté.

- *Un exemple avec options :*

```
proc logistic data=malib.employe outest=param noprint;
 class poste diplome sexe ;
 model prime=sexe diplome poste anciennete subor /selection=stepwise ;
run;
```

L'option noprint supprime la sortie Output. L'option outest permet d'obtenir la table param suivante :

LINK	_TYPE_	_STATUS_	_NAME_	Intercept	sexeF	diplome0	diplome1	diplome2	diplome3	diplome4	_manager postecadre	_technicien postecadre	poste maintenance	anciennete	subor	_LNLIKE_
LOGIT	PARMS	0 Converged	prime	6,081106	-1,358777	-1,748812	-0,154109	5,243927	2,709546	-0,967802				-0,483097		-25,68922

L'option de sélection aboutit au modèle constitué des variables sexe, diplôme et ancienneté. On a récupéré les valeurs des paramètres estimés, ainsi que certains renseignements comme la fonction de lien, le nom de la variable, le résultat du test de convergence et le log de la vraisemblance.

Modélisation à plusieurs équations avec la PROC SYSLIN et la PROC MODEL

La procédure PROC SYSLIN a pour but de réaliser des estimations et des prédictions sur des systèmes d'équations linéaires. Son équivalent pour des systèmes d'équations non linéaires est la PROC MODEL.

Ces deux procédures offrent un large choix de méthodes d'estimation :

- moindres carrés ordinaires (OLS) (mais on sait que cet estimateur est biaisé dans le cas d'un système d'équations),
- estimateur SURE (méthode simple ou itérative),
- doubles moindres carrés (2SLS),
- triples moindres carrés (3SLS) (méthodes simples ou itératives),
- maximum de vraisemblance à information limitée (LIML)
- maximum de vraisemblance à information complète (FIML),…

Elles offrent également la possibilité :

- d'imposer des contraintes sur les paramètres estimés
- de tester des hypothèses de liaison linéaire entre les paramètres

La lecture d'une sortie standard de PROC SYSLIN est comparable à celle d'une PROC REG. La modélisation de chaque équation du système est présentée sur une page de l'output.

```
PROC SYSLIN  data=   <options> ;
      < ENDOGENOUS variables ; >
      < INSTRUMENTS variables ; >
      MODEL dep_var1 = liste_des_régresseurs </options>;
      MODEL dep_var2 = …
      < RESTRICT équations ; >
      < SRESTRICT équations ; >
      < TEST équation ; >
      < STEST équation ; >
      < BY variable ; >
      < WEIGHT variable ; >
```

Les instructions :

Il y a autant d'instructions MODEL que d'équations dans le système estimé.

Les instructions ENDOGENOUS et INSTRUMENTS vont ensemble. Elles ne sont nécessaires que si la méthode d'estimation n'est pas une des suivantes : OLS, SUR, ITSUR ou FIML. Les autres méthodes d'estimation effectuent d'abord une régression des variables endogènes sur les variables instrumentales du modèle. L'instruction INSTRUMENTS sert donc à lister les variables instrumentales du modèle. Les variables listées par ENDOGENOUS sont donc les variables du modèle qui ne sont pas instrumentales.

Comme dans la PROC REG, on peut imposer des restrictions ou effectuer des tests. Les restrictions et tests ne concernant qu'une des équations du modèle sont spécifiés respectivement par les instructions RESTRICT et TEST (pour plus de détails sur ces instructions, voir le paragraphe relatif à la PROC REG). Une restriction ou un test faisant intervenir des paramètres de différentes équations utilisent les instructions SRESTRICT et STEST.

L'instruction BY sert toujours à définir des sous-populations.

L'instruction WEIGHT sert toujours à définir une variable de pondération pour les observations.

Exemple :

```
PROC SYSLIN ;
 ENDOGENOUS y1 y2 ;
 INSTRUMENTS x1 x2 ;
 MODEL y1 = y2 x1 x2 ;
 MODEL y2 = x1 y1 ;
 SRESTRICT y1.x1=y2.x1 ;
RUN ;
```

Les options :

En option de la procédure, on peut demander à récupérer entre autres :

- les paramètres estimés : option outest=
- l'ensemble des résidus et des paramètres estimés : option out= , utilisée avec une instruction `OUTPUT PREDICTED=`*`nom_vars`* `RESIDUAL=`*`nom_var`* `;`

➢ *Les options précisant la méthode d'estimation :*

ols	estimateur des moindres carrés ordinaires
sur	estimateur par la méthode SURE
itsur	méthode SURE itérative
fiml	maximum de vraisemblance à information complète
liml	maximum de vraisemblance à information limitée
2sls	estimateur des doubles moindres carrés
3sls	estimateur des triples moindres carrés
it2sls	doubles moindres carrés itératifs
it3sls	triples moindres carrés itératifs
melo	méthode du minimum de perte espérée
k=*valeur*	méthode de la k-class

Enfin mentionnons l'option first qui édite la régression des endogènes sur les instruments.

➢ *Les options de l'instruction MODEL :*

Concernant les options de l'instruction MODEL, outre deux classiques (noint qui permet d'éliminer la constante du modèle et dw qui réalise un test de Durbin Watson), nous mentionnerons plot qui édite des graphiques des résidus en fonction de chacun des régresseurs (un graphique par régresseur).

Quoi de neuf avec la PROC MODEL ?

Cette fois l'estimation ne porte pas forcément sur des équations linéaires. L'instruction MODEL, propre à la régression linéaire, disparaît donc.

Les équations sont écrites telles qu'on les écrit naturellement, dans le corps de la procédure. Une instruction PARAMETERS permet de préciser quelles grandeurs jouent le rôle des paramètres à estimer.

On retrouve les instructions ENDOGENOUS et INSTRUMENTS, ainsi qu'une instruction EXOGENOUS qui vient les compléter. Par défaut, toutes les variables exogènes sont considérées comme instruments. On n'utilisera donc l'instruction INSTRUMENTS que si ce n'est pas le cas.

Enfin, on précisera quelles sont les variables à régresser grâce à une instruction FIT. Il est à noter que c'est en option de l'instruction FIT que l'on précisera la méthode d'estimation. Notons que la liste des méthodes disponibles ne coïncide pas tout à fait avec celle de la PROC SYSLIN.

Les instructions RESTRICT et TEST sont encore valides.

Pour plus de détails, on se réfèrera à l'aide de SAS.

Partie II C

Procédures graphiques

La PROC GCHART trace des graphiques sur des variables qualitatives : diagrammes en bâtons ou diagrammes circulaires.

La PROC GPLOT trace des graphiques sur des variables quantitatives : nuages de points, courbes.

La PROC BOXPLOT trace des box plots (boîtes à moustaches) représentatives de la distribution d'une variable sur plusieurs sous-populations.

Graphiques pour variables qualitatives avec la PROC GCHART

La procédure PROC GCHART permet de réaliser des graphiques tels que des diagrammes en bâtons ou des diagrammes circulaires (« camemberts »), ou encore des diagrammes en blocs ou en étoiles, comme ceux réalisés par la PROC CHART mais en plus joli. Elle peut aussi réaliser des graphiques en 3D. Notons que cette procédure est incluse dans l'outil presse bouton Graph n Go (voir le paragraphe consacré dans la partie III B).

```
PROC GCHART data=  <options> ;
      HBAR variables </options> ;
      VBAR variables </options> ;
      BLOCK variables </options> ;
      STAR variables </options> ;
      PIE variables </options> ;
      HBAR3D variables </options> ;
      VBAR3D variables </options> ;
      PIE3D variables </options> ;
      < BY variables ; >
```

Les instructions :

Les cinq premières instructions proposées ne sont pas toutes obligatoires. Il en faut juste au moins une. Elles servent à définir la liste des variables qualitatives sur lesquelles on souhaite faire des graphiques, ainsi que le type de graphique souhaité.

L'instruction HBAR réalise des diagrammes en bâtons horizontaux, un par variable spécifiée. L'instruction VBAR réalise des diagrammes en bâtons verticaux.
L'instruction BLOCK réalise des diagrammes par blocs et l'instruction STAR des diagrammes en étoiles.
Enfin, l'instruction PIE réalise des diagrammes circulaires (camemberts).
Les trois autres instructions parlent d'elles mêmes.

L'instruction BY sert toujours à définir des sous-populations. Dans ce cas, un graphique différent est tracé pour chacune de ces sous-populations.

Les options :

L'option gout= permet d'indiquer un catalogue dans lequel sauvegarder les graphiques (voir à ce sujet la partie III C).

- *Quelques options des instructions HBAR / HBAR3D / VBAR / VBAR3D :*

L'option ascending (resp. descending) permet d'ordonner les bâtons dans l'ordre croissant (resp. décroissant) de leur taille.

L'option nozero permet de ne pas éditer les bâtons correspondant à une occurrence nulle de la modalité. Cela peut servir notamment en présence d'une instruction BY, lorsque les sous-groupes formés sont hétérogènes.

L'option group= et l'option subgroup= permettent de faire des sous-groupes selon les modalités d'une certaine variable. L'option group= génère des diagrammes juxtaposés tandis que l'option subgroup= génère des diagrammes superposés.

L'option midpoints= permet de définir les valeurs qui doivent être au centre des blocs. Cela sert si la variable est numérique.

L'option outside= place des statistiques au-dessus des barres (seulement dans le cas d'un histogramme vertical). Cela peut-être : la fréquence (mot-clé freq), la fréquence cumulée (mot-clé cfreq), le pourcentage (mot-clé percent), le pourcentage cumulé (mot-clé cpercent), la somme (mot-clé sum) ou la moyenne (mot-clé mean).

L'option type= permet de préciser l'ordonnée du graphique. Par défaut, il s'agit de la fréquence. Mais on peut préférer la fréquence cumulée, le pourcentage ou le pourcentage cumulé. On peut également choisir de calculer la somme ou la moyenne d'une variable relatives aux différentes modalités

envisagées. On ajoute alors une option sumvar= . Par exemple, le graphique que l'on souhaite obtenir doit représenter le salaire moyen par catégorie socio-professionnelle : alors csp est dans VBAR, et on précise type=mean et sumvar=salaire.

Il existe aussi de nombreuses options de mise en forme qu'on ne détaille pas ici.

- *Quelques options des instructions PIE / PIE3D :*

L'option explode=*nom_modalité* permet de faire ressortir une part du camembert pour la mettre en valeur.

L'option other=*p* permet de regrouper les modalités dont le pourcentage est inférieur à *p* dans une catégorie autres.

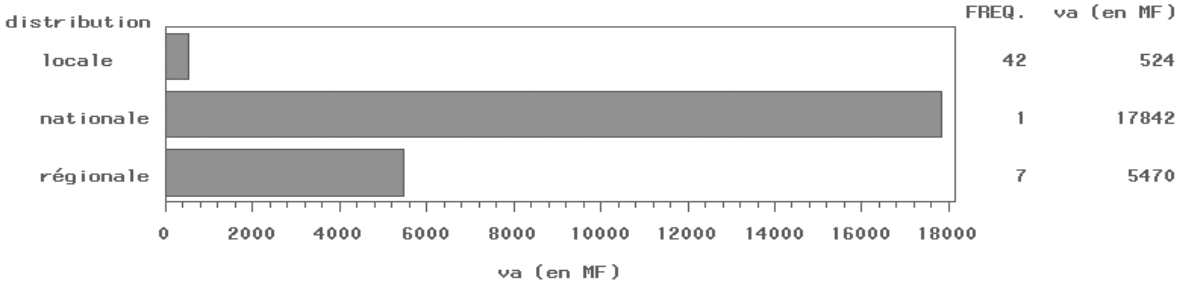
On peut encore utiliser l'option subgroup=. Il est alors possible d'utiliser l'option legend qui crée des légendes (afin de ne pas surcharger le diagramme).

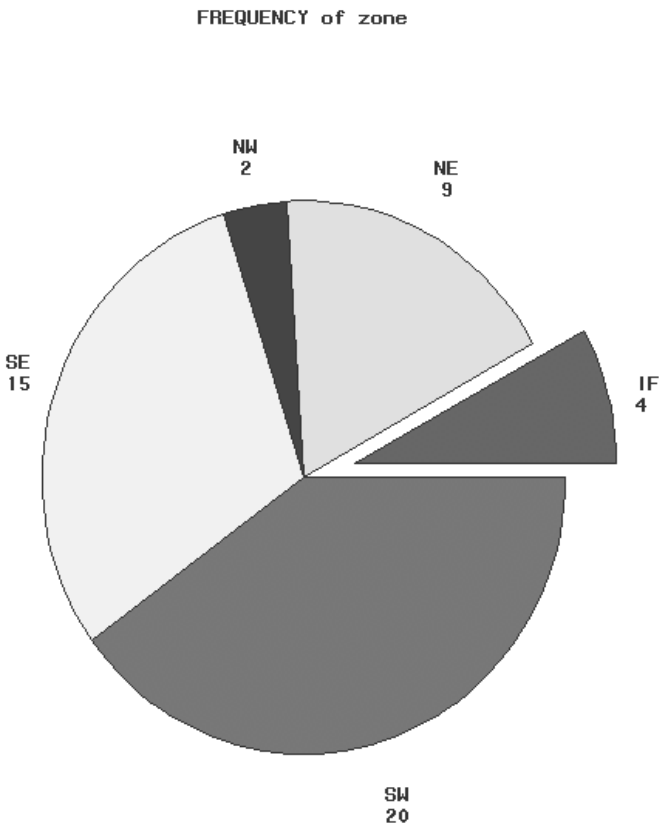
Les options midpoints= , ascending / descending, type= et sumvar= sont encore valables.

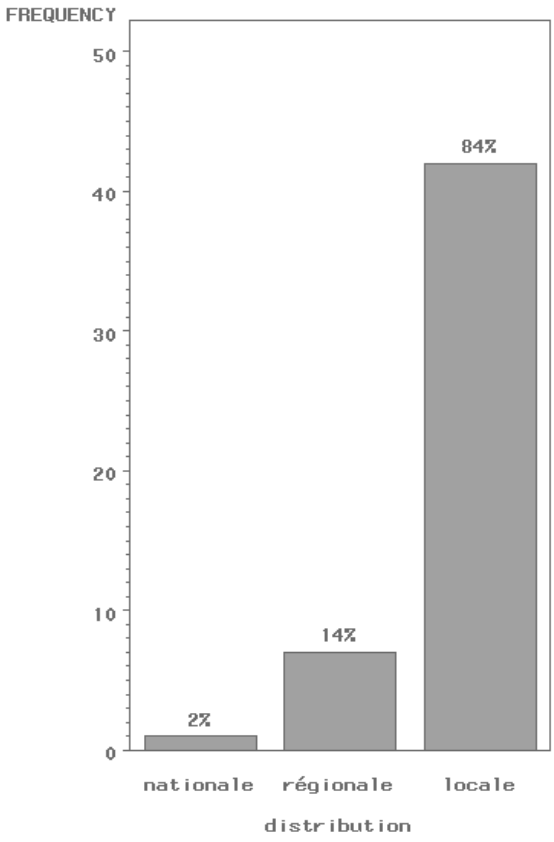
Il existe aussi de nombreuses options de mise en forme pour lesquelles on se reportera à l'aide en ligne de SAS.

Exemple :

```
proc gchart data=malib.bidon ;
  hbar distribution / sumvar=va type=mean ;
  pie zone / explode='IF' ;
  vbar distribution / ascending outside=percent ;
run ;
```




FREQUENCY of zone
NW
2
NE
9
SE
15
IF
4
SW
20


FREQUENCY
50
40
30
20
10
0
84%
14%
2%
nationale
régionale
locale
distribution

Graphiques pour variables quantitatives avec la PROC GPLOT

La procédure PROC GPLOT permet de réaliser des graphiques qui croisent deux variables, comme ceux réalisés par la PROC PLOT mais en plus joli.
Notons que cette procédure est incluse dans l'outil presse bouton Graph n Go (voir le paragraphe consacré dans la partie III B).

```
PROC GPLOT data=  <options> ;
      PLOT  yvar * xvar <=points_var><="caractère"> </options> ;
      < PLOT2 yvar * xvar ; >
      BUBBLE  yvar * xvar = bubble_var  </options> ;
      < BY variables ; >
      < SYMBOL <interpol= > ; >
```

Les options :

L'option uniform indique (s'il y a une instruction BY) que tous les graphes doivent avoir la même échelle.

L'option gout= permet d'indiquer un catalogue dans lequel sauvegarder les graphiques (voir à ce sujet la partie III C).

Les instructions :

L'instruction PLOT permet de spécifier les variables à mettre en ordonnée et en abscisse. La syntaxe est la suivante :

```
PLOT ordonnée*abscisse ;
```

On peut demander plusieurs graphiques dans la même instruction PLOT, par exemple :

```
PLOT y*x1 y*x2 ;
```

Auquel cas on peut « factoriser » y :

```
PLOT y*(x1 x2) ;
```

L'instruction PLOT peut être remplacée par une instruction BUBBLE. L'instruction BUBBLE crée un graphique à trois variables : une en abscisse, une en ordonnée, et une définissant la taille des « bulles ». Notons que pour ces deux instructions la première des variables indiquées sera en ordonnée. Notons encore que *bubble_var* doit être numérique.

A l'instruction PLOT on peut aussi adjoindre une instruction PLOT2. Cela permet de superposer des graphiques avec des échelles différentes.

L'instruction SYMBOL sert à définir l'aspect du graphique.
A l'intérieur de cette instruction, le mot-clé interpol= permet de préciser si les points doivent être reliés et la manière dont ils le sont.

- scatter : nuage de points (option par défaut)
- line : segments
- spline : courbe
- regression : droite de régression
- needle : chaque point est relié à son abscisse
- join : linéaire par morceaux
- l : interpolation de Lagrange
- step : fonction en escalier

Toujours à l'intérieur de l'instruction SYMBOL, on peut écrire : pointlabel= « #var » ; dans ce cas, SAS étiquette les points avec la valeur correspondante de la variable var.
Exemple :
L'instruction `SYMBOL interpol=spline pointlabel= « #lab » ;` commande que les points soient reliés entre eux par une courbe et étiquetés par la valeur de la variable lab.

D'autres instructions existent encore pour la mise en page des graphes.

Quelques options des instructions PLOT et BUBBLE :

Dans l'instruction PLOT yvar*xvar, si l'on rajoute = « * », les points apparaitront sous forme d'étoiles. Si l'on rajoute =var, où var est qualitative avec peu de modalités, les points prennent la forme des modalités de var.

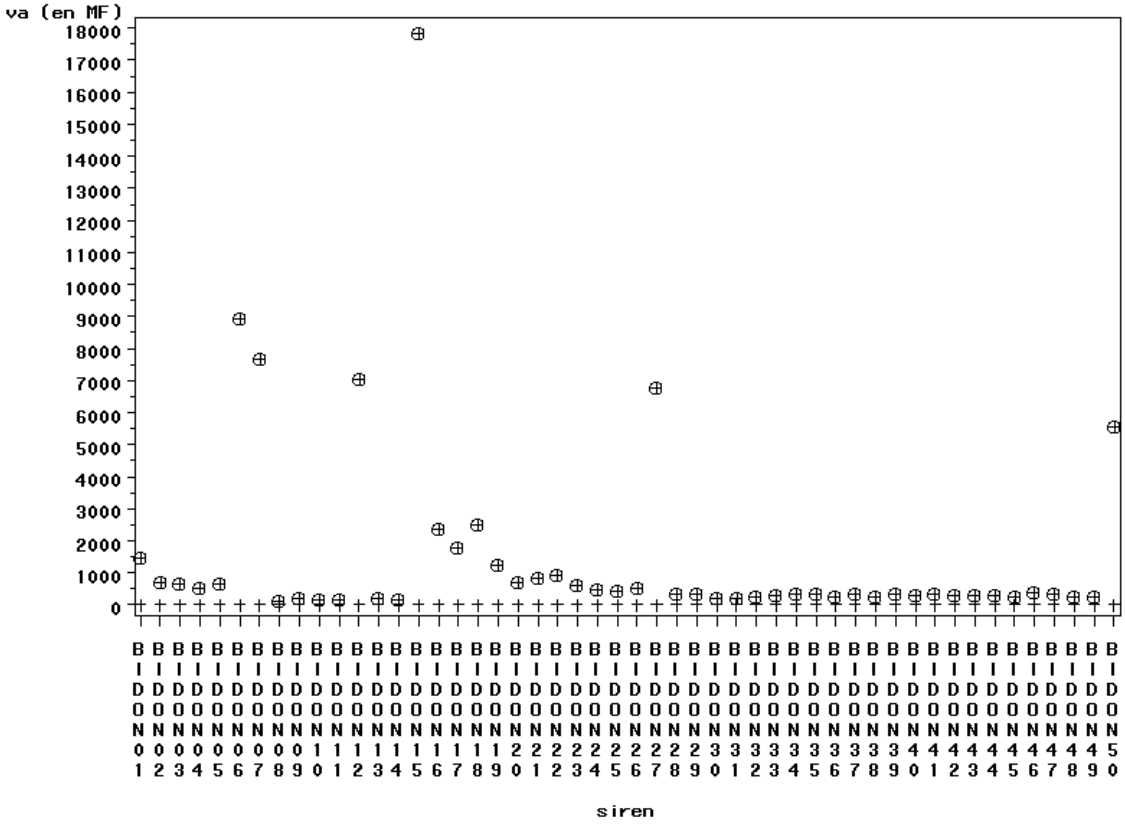
L'option overlay permet de superposer des plots.

Si on a demandé la droite de régression, l'option regeqn édite l'équation de la régression en question.
Mis à part cela, les options de l'instruction PLOT sont des options de mise en page (quadrillage, couleurs, axes, légende…)

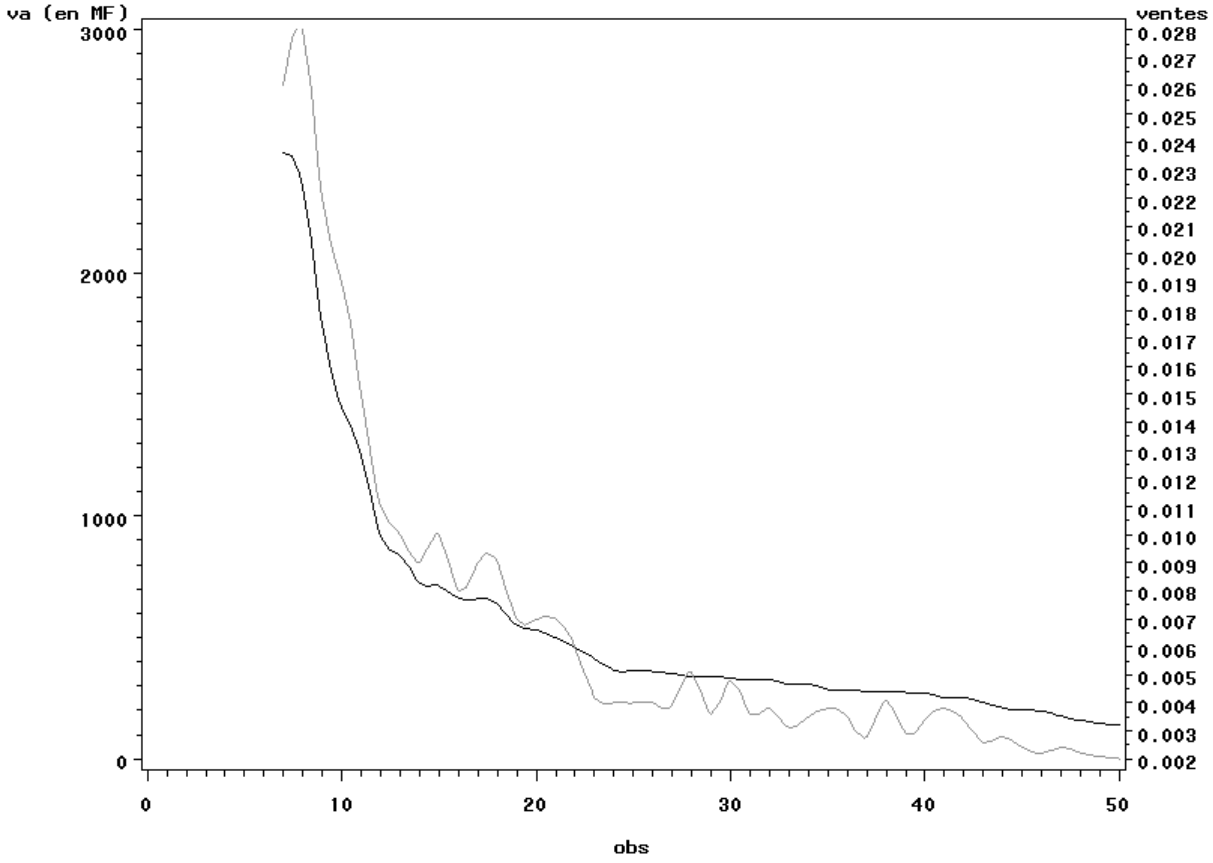
L'option blabel édite les valeurs de *bubble_var* à côté de chaque bulle. Les options bsize= et bcolor= permettent respectivement de choisir la taille et la couleur des bulles. Les autres options sont communes avec PLOT.

Exemples :

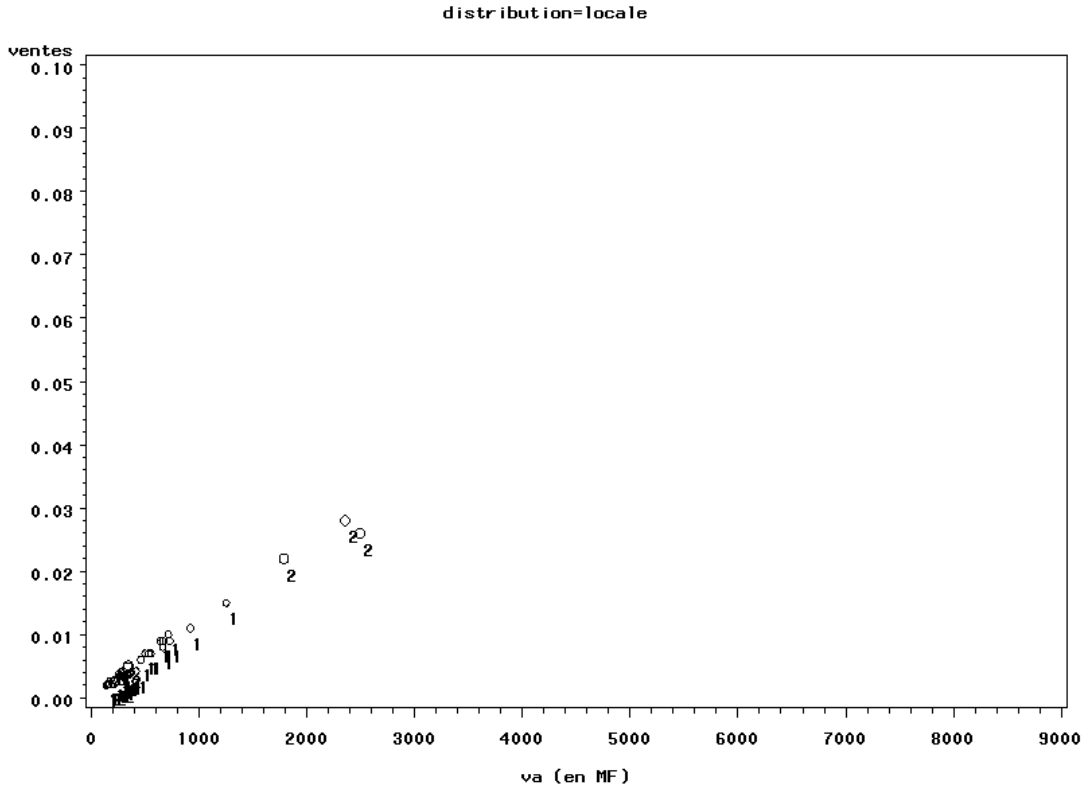
```
proc gplot data=malib.bidon;
 plot va*siren="+" ventes*siren="o" /overlay;
run;
```

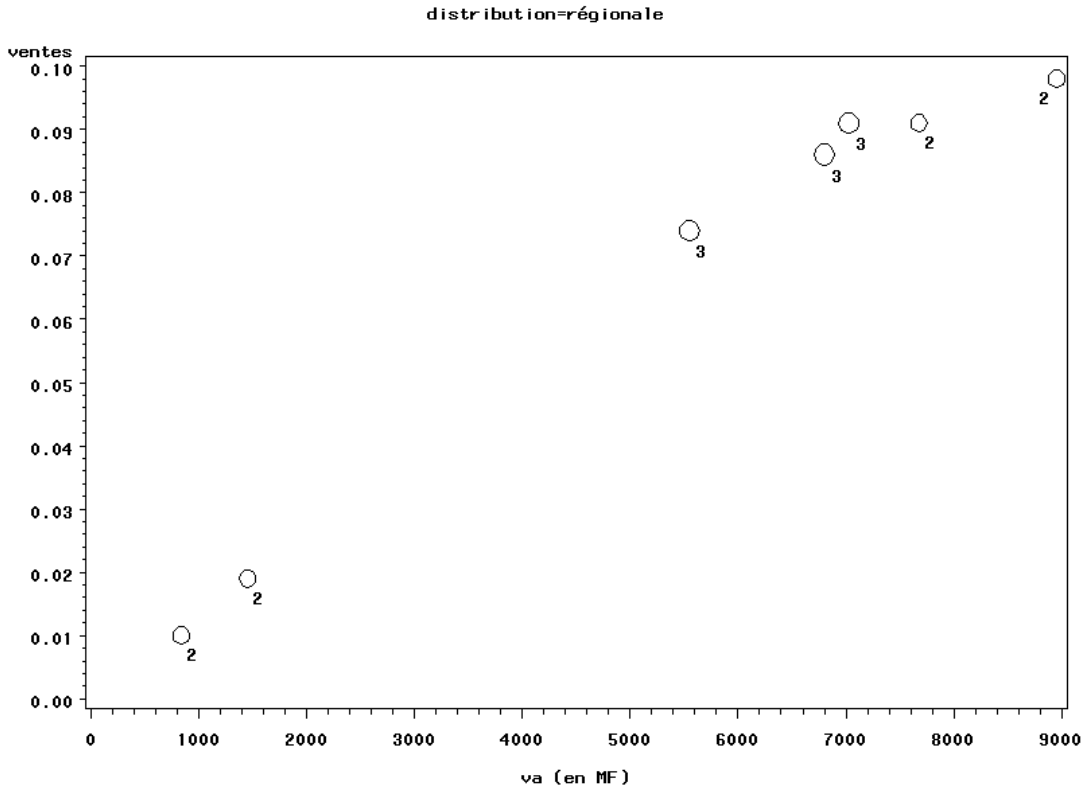


```
proc gplot data=bidon;
 plot va*obs ;
 plot2 ventes*obs ;
 symbol interpol=spline;
 where va<3000;
run;
```



```
proc gplot data=bidon uniform;
 bubble ventes*va=taille2 /blabel bsize=3;
 by distribution;
run;
```





Des boîtes à moustaches avec la PROC BOXPLOT

Cette procédure permet de tracer des diagrammes de distribution appelés « boîtes à moustaches » ou « box plots », en juxtaposant les diagrammes relatifs à plusieurs sous-populations.

```
PROC BOXPLOT data=  <options> ;
      PLOT  var * groupvar </options> ;
      < INSET mots_clés_statistiques ; >
      < INSETGROUP mots_clés_statistiques ; >
```

Les instructions

L'instruction PLOT permet de définir :
var : La variable d'intérêt quantitative continue pour laquelle on veut tracer les boxplots.
groupvar : La variable définissant les sous-groupes de population.

Les instructions INSET et INSETGROUP permettent d'insérer sur le graphique des encadrés de statistiques. L'instruction INSET calcule ces statistiques sur toute la population, tandis que INSETGROUP les calcule sur chacun des sous-groupes.

- *Statistiques accessibles avec INSET :*

MEAN	Moyenne
MIN	Valeur minimale de la variable étudiée
MAX	Valeur maximale de la variable étudiée
NMIN	Nombre d'observations minimal
NMAX	Nombre d'observations maximal
NOBS	Nombre total d'observations
STDDEV	Ecart-type de la variable étudiée

- *Statistiques accessibles avec INSETGROUP :*

MEAN	Moyenne
MIN	Minimum
MAX	Maximum
N	Nombre d'observations
NHIGH, NLOW et NOUT	Nombre d'individus atypiques (respectivement valeurs hautes, basses, et toutes)
Q1, Q2 et Q3	Quartiles
RANGE	Max - Min
STDDEV	Ecart-type

Les options

L'option gout= permet d'indiquer un catalogue dans lequel sauvegarder les graphiques (voir à ce sujet la partie III C).

- *Les options de l'instruction PLOT :*

L'option outbox= permet de récupérer dans une table la valeur de statistiques pour chaque groupe ainsi que les valeurs atypiques.

PARTIE III

Aide à l'utilisation du logiciel

Partie III A

Conseils pratiques

Cette partie fournit quelques conseils pour une utilisation plus fluide du logiciel :

Un exemple d'utilisation de l'aide.
A retenir :

- Utiliser plutôt l'onglet Index, car l'onglet Recherche fournit des résultats pléthoriques.
- Taper en premier les mots les plus significatifs : « FREQ Procedure » et non « PROC FREQ ».

Un recensement des messages d'erreur les plus fréquemment rencontrés, avec leur diagnostic.

Deux options pour réduire la taille de la LOG, et donc mieux s'y retrouver dans les messages.

Enfin, un « kit de secours » ou : que faire en cas de plantage léger ou aigu du logiciel ?

Exemple d'utilisation de l'aide de SAS

L'aide de SAS est hiérarchisée. Supposons pour reprendre l'exemple précédent que l'on recherche comment ne pas éditer les colonnes cumulées dans un tableau de fréquence. Voilà comment l'on procèdera pour être le plus efficace dans ce cas :

- Dans la partie index de l'aide SAS System Help.
- On tape FREQ (c'est le nom de la procédure).
- Dans la liste, on choisit FREQ Procedure.
- Eventuellement on clique sur le lien Syntax.
- La syntaxe générale de la PROC FREQ apparaît. Généralement, la PROC ainsi que toutes les instructions constituent des liens. Cliquer sur ce lien permet d'accéder au détail de l'instruction (ce qu'elle fait, la forme à donner aux arguments) ainsi qu'aux options disponibles.
- On clique donc sur le lien TABLES, car on se doute que l'option que l'on recherche se rattache à cette instruction.
- C'est dans cette page que l'on trouve l'option nocum. Victoire !

Réservez l'usage de la Recherche pour les cas où vous n'arrivez vraiment pas à trouver par l'index. Elle fournit en effet des résultats prolifiques et non ordonnés, et il est difficile de s'y retrouver.

Les erreurs les plus fréquentes recensées dans la LOG

La plupart des erreurs donnant lieu à des messages dans la LOG consistent en des erreurs de syntaxe aisément réparables.

Le mot sur lequel le programme bloque est presque toujours souligné en rouge. L'erreur tient alors soit à ce mot soit à une syntaxe incomplète placée juste avant.

Souvent, il suffit de lire le message attentivement, car il est assez explicite. Lorsqu'il ne l'est pas, commencez par chercher le point virgule qui manque !

➢ *Quelques exemples de messages fréquemment rencontrés :*

`No matching DO/SELECT statement`

On a mis un END; autrement que pour fermer un DO; ou un SELECT;

`There was 1 unclosed DO block`

On a oublié le END;

`The variable m in the DROP, KEEP, or RENAME list has never been referenced`

Soit il y a une faute de frappe dans le nom de la variable soit elle n'existe pas dans la table considérée soit il y a eu un rename plus haut dans l'étape data qui fait qu'elle a changé de nom.

`Libname malib is not assigned`

La librairie malib n'existe pas

`Variable o is uninitialized.`
`Missing values were generated as a result of performing an operation on missing values.`

Ces deux messages (en bleu) sont délivrés ensemble lorsqu'une variable est utilisée à laquelle aucune valeur n'est affectée. Là encore il faut penser à une éventuelle faute de frappe, par exemple o au lieu de 0.

`You cannot open WORK.FINALE.DATA for output access with member-level control because WORK.FINALE.DATA is in use by you in resource environment ViewTable Window`

La table table dans la librairie work est ouverte donc on ne peut pas l'utiliser dans le programme soumis.

`Statement is not valid or it is used out of proper order`

Soit l'instruction appelée n'existe pas ou est mal orthographiée, soit elle n'est pas appropriée dans le contexte donné, soit il manque un ; avant son appel.

`Data set was not specified on the DATA statement`

Le mot qui suit l'instruction output ne correspond à aucune des tables créées dans l'étape data.

`Undeclared array referenced: x`

La notation x(i) a été utilisée dans un cas autre que celui où x est un tableau (défini par l'instruction ARRAY). Soit x est une variable et alors on oublie cette notation, soit x est une fonction SAS et alors son résultat doit être récupéré dans une variable (on écrit a=x(0) ; et non x(0) ;)

`Expecting a ;`

A cet endroit il faut un point virgule.

`Syntax error, expecting one of the following:...`

Soit il manque un ; ou une parenthèse ou un caractère de ce style, soit les options ou instructions spécifiées ne sont pas utilisées à bon escient.

`User does not have appropriate authorization level for file MALIB.COURS.DATA.`

La librairie malib est assignée sur un répertoire sur lequel on n'a pas suffisamment de droits d'accès.

Comment diminuer le contenu de la log SAS ?

L'option nonotes permet de supprimer les notes SAS dans la LOG. Les messages d'alerte (Warning) et messages d'erreur sont toujours visibles.
L'option noechoauto permet de ne pas afficher dans la LOG le résultat de l'exécution d'un fichier autoexec.sas

Kit de secours

Lorsqu'un programme semble ne pas devoir finir, le premier réflexe à avoir est de cliquer sur l'icône « Stop » ! Le logiciel peut mettre un peu de temps à réagir, mais cette solution fonctionne la plupart du temps. Une pop-up s'ouvre alors, vous proposant soit de quitter SAS, soit d'interrompre les processus en cours. C'est bien sûr cette seconde proposition qu'il faut choisir.

Dans d'autres cas, le programme semble s'être terminé correctement, mais le logiciel se met à avoir un comportement hératique, reconnaissable par exemple à l'absence de message dans la LOG, ou bien des messages avertissant que les données sont en cours d'utilisation. Ces erreurs font souvent suite à un bug antérieur ou à l'appel incorrect d'une procédure. On peut alors essayer le remède miracle suivant, qui consiste à taper `quit ;` dans l'editor et à soumettre.

Dans tous les cas, évitez d'ouvrir plusieurs sessions SAS en même temps, car le logiciel le supporte mal.

Partie III B

Des assistants pour gagner du temps

L'outil Graph 'n Go

Le module **Solutions – Reporting – Graph 'n Go** ouvre un assistant "presse-bouton" qui fait appel au module SAS/GRAPH. En particulier, il permet de paramétrer en quelques clics des procédures de type GCHART ou GPLOT.

Première étape : Choix de la source de données

La première icône en haut à gauche représente une table SAS. Cliquez dessus pour choisir la table contenant les données source de vos graphiques.
SAS affiche les propriétés de la table. Il est possible à ce niveau d'introduire un filtre sur les colonnes.

Une fois la table choisie, une icône la représentant apparaît dans le corps principal de la fenêtre.
Par un clic droit – Properties, il est possible de paramétrer vos données, par exemple de filtrer les observations selon une condition where (onglet « subset data »).

Deuxième étape : Choix du type de graphique

Les autres icônes représentent chacune un type de graphique : diagramme en bâtons (équivalent de la proc gchart avec instruction hbar ou vbar) ; diagramme circulaire (équivalent de la proc gchart avec instruction pie) ; courbe (équivalent de la proc gplot) ; courbe superposée avec un histogramme.
A noter l'icône marquée d'un « A » qui permet d'insérer des zones de texte.
Choisissez le type de graphique voulu en le faisant glisser vers le corps principal de la fenêtre (simple clic, glisser, lâcher, puis simple clic pour valider l'emplacement). Un cadre est dessiné avec comme titre le type de graphe.

Troisième étape : Paramétrage du graphique

- *Onglet Data :*

Par un clic droit – Properties, vous ouvrez la fenêtre de paramétrage du graphique.
Le premier onglet qui se présente est l'onglet « Data », qui permet de sélectionner la source des données parmi les tables sélectionnées dans la première étape.
Ensuite seulement vous pouvez indiquer au logiciel les variables qu'il doit considérer en abscisse et en ordonnée.

Diagramme en bâtons :

Category : Il s'agit de la variable qualitative pour laquelle vous souhaitez réaliser le diagramme en bâtons ; en d'autres termes, l'abscisse du diagramme.
Response : Par défaut –NONE- qui signifie que la hauteur des bâtons sera déterminée par les effectifs (réels ou en pourcentage) de chacune des modalités de la variable « category ». Il est possible de choisir aussi une variable quantitative de la table source, dans ce cas la hauteur des bâtons dépend des valeurs de cette variable (en somme ou en moyenne).
Group : Variable dont les modalités définissent des sous-populations ; les bâtons seront juxtaposés.
Subgroup : Variable dont les modalités définissent des sous-populations ; les bâtons seront superposés.
Statistic : Frequency par défaut (effectifs) ; on peut aussi choisir Percent, Cumulative frequency ou Cumulative Percent. Lorsqu'une variable est indiquée en « response », la statistique par défaut est la moyenne (Average), mais on peut aussi choisir la somme, la part dans la somme ou encore la part cumulée dans la somme.

Enfin, il est possible d'afficher la valeur de l'ordonnée en cochant « **Show bar values** ».

Diagramme circulaire :

Category : Il s'agit de la variable qualitative sur laquelle vous souhaitez réaliser le diagramme circulaire.
Response : Par défaut –NONE- qui signifie que la taille des secteurs est déterminée par les effectifs (réels ou en pourcentage) de chacune des modalités de la variable « category ». Il est possible de choisir aussi une variable quantitative de la table source, dans ce cas la taille des secteurs dépend des valeurs de cette variable (en somme ou en moyenne).
Statistic : Frequency par défauts (effectifs), mais on peut aussi choisir Percent (pourcentages). Lorsqu'une variable est indiquée en « response », la statistique par défaut est la moyenne (Average), mais on peut aussi choisir la somme (Sum).

Nuage de points / courbes :

X : Choix de la variable à placer en abscisse.
Y : Choix de la variable à placer en ordonnée.
ID : Choix éventuel d'une variable dont les valeurs vont « étiqueter » les points.
Plot style : Type de graphique, par défaut Scatter (nuage de points). On peut aussi choisir Join (les points sont reliés par des segments), Needle (les points sont reliés à l'axe des abscisses), Step (les points sont reliés par une fonction en escaliers), Spline (les points sont reliés par une courbe) ou encore Regression (une droite de régression est tracée pour le nuage de points).

➢ *Onglet Titles / Footnotes :*

Permet de définir un titre et éventuellement un pied de page au graphique.

➢ *Onglet Appearance :*

Permet de paramétrer l'apparence du graphique.

Diagramme en bâtons :

Color scheme : Choix des couleurs.
Reference Lines : Trace des lignes de « référence » perpendiculaires aux bâtons.
Bar style : Choix du style des bâtons.
Orientation Vertical / Horizontal : Choix de l'orientation des bâtons.
Visible Midpoints : Par défauts, toutes les modalités sont visibles et correspondent chacune à un bâton, mais on peut modifier cela.
Order Midpoints by Category value / Statistic value : Choix du tri des bâtons, soit par ordre des valeurs de la variable étudiée, soit par ordre de grandeur des bâtons (ascendant ou descendant).

Diagramme circulaire :

Color scheme : Choix des couleurs.
Legend : Affichage de la légende si cette option est cochée.
Pie style : Permet de choisir le style de camembert (normal ou 3D).
Orientation : Permet de définir le point de départ des tranches (par défaut, à 3h).
« Other » label : Permet de définir un libellé pour la modalité « autres ».

Nuage de points / courbes :

Color scheme : Choix des couleurs.
Plot style : Choix du style de graphe (voir onglet data).
Reference Lines X axis / Y axis : Trace des lignes de référence parallèles respectivement à l'axe des ordonnées et à l'axe des abscisses.
Point markers : Choix de l'apparence des points ainsi que de leur taille.

Quatrième étape: Gestion de mes graphiques

Un clic droit sur un graphique permet d'accéder à un certain nombre d'options :

- Copie, suppression, déplacement du graphique dans la fenêtre.
- Réduction / Agrandissement du graphique. Notamment, l'option **Maximize** permet d'afficher le graphique en pleine fenêtre. A noter que toute opération est interdite tant que le graphique est dans cet état.
- Exportation du graphique (voir à ce sujet la partie IIIC).
- Accès aux propriétés du graphique.

SAS Query

Le menu **Tools – Query** ouvre un assistant "presse-bouton" qui fait appel au module SAS/SQL. En d'autres termes, il permet de faire des PROC SQL sans connaître le SQL.

Les étapes de la PROC SQL avec Query

Première fenêtre : il s'agit de choisir la librairie, puis les tables sur lesquelles on veut travailler (sélectionner dans la colonne de gauche puis faire glisser à droite grâce à la petite flèche. Si plusieurs tables sont sélectionnées, Query comprend qu'il faut faire une jointure de ces différentes tables.

Deuxième fenêtre : choix des variables à garder parmi les variables des tables sélectionnées.
La colonne du milieu présente plusieurs possibilités :

Column Alias / Label : permet d'affecter un alias (= un autre nom) ou un label à une colonne

Column Formats : permet de choisir le format de la colonne

Summary functions : permet de calculer le count ou la somme, la moyenne, le nombre de valeurs manquantes,… de le colonne

Move before et **Move after** : pour réorganiser l'ordre des colonnes

Build a column : c'est le seul qui n'ait pas besoin qu'une colonne soit sélectionnée pour fonctionner. Permet de construire une nouvelle colonne.

> ➢ *A partir de là, beaucoup de menus sont accessibles par un clic droit dans la fenêtre de droite.*

Where conditions for subset : pour rajouter des conditions de type WHERE.
Apparaît alors une troisième fenêtre. A droite on clique sur la colonne concernée. Query propose alors une liste d'opérateurs. Enfin on complète la condition.
On remarque qu'on peut entrer une valeur constante en cliquant sur <CONSTANT enter value> et qu'on peut visualiser la liste des modalités de la variable choisie en cliquant sur <LOOKUP distinct values>. Ces deux possibilités se trouvent dans la fenêtre de droite comme la liste des variables.
Enfin mentionnons qu'on peut entrer plusieurs conditions WHERE en les reliant par un opérateur choisi grâce à l'onglet OPERATORS.

Order by : pour trier le résultat de la requête

Groups for summary functions : équivalent du GROUP BY,

Having condition for group : équivalent de la ligne HAVING complétant le GROUP BY,

Join type : permet de choisir le type de jointure et la variable sur laquelle on fait la jointure (par défaut on fait des jointures internes = « inner »). Dans le premier des programmes ci-dessus, on fait une jointure interne sur la variable nom : on met sur la même ligne les informations ayant trait à la même personne et on ne garde que les noms qui apparaissent dans les deux tables. Une jointure externe au contraire garde toutes les modalités de nom et complète la table avec des valeurs manquantes.

Reset permet de réinitialiser la requête.

Tables permet de modifier les tables sélectionnées.

Run query permet de soumettre le programme Query.

Show query permet de visualiser le programme généré par Query. On peut également à partir de là choisir de sauver le résultat de la requête dans une table, ou bien enregistrer la requête dans un catalogue.

PARTIE III C

Sauvegarde et exportation des sorties SAS

Le copier-coller de l'Ouput vers un document Word peut rapidement inspirer des envies de meurtre tant c'est laid, fastidieux et capricieux. Pourquoi s'embêter, alors que les sorties des procédures SAS, qu'elles se trouvent dans la fenêtre Output ou dans une autre (fenêtre Graph pour les graphiques obtenus avec les proc gplot et gchart, fenêtre Insight…) peuvent être sauvegardées ?

On distingue trois grandes catégories de sauvegarde :

La sauvegarde sous forme d'un objet SAS dans un catalogue.

Ce type de sauvegarde peut s'appliquer aux sorties Output, aux formats compilés, aux graphiques qui se trouvent dans une fenêtre Graph ou à ceux obtenus avec l'outil Graph 'n Go, aux tableaux Insight, aux requêtes créées avec SAS Query…
C'est la méthode la plus pratique pour l'échange d'objets SAS entre membres d'un groupe de travail : un seul fichier sas7bcat peut contenir tous les objets, et on n'a pas à mettre le code source à disposition…

La sauvegarde en tant que fichier texte ou fichier image.

Les sorties Output peuvent aussi s'enregistrer en tant que telles, sous une extension spécifique à SAS (.lst). On peut ensuite les ouvrir sous SAS ou dans un bloc-notes.
Les graphiques et les sorties Insight peuvent être exportés sous format image. Les inconvénients de cette méthode sont la taille des fichiers obtenus et l'impossibilité de les modifier.

L'exportation via l'Output Delivery System (ODS).

L'ODS permet d'exporter n'importe quelle sortie de PROC sous un format lisible avec Word : HTML ou RTF. Les sorties exportées sont présentées d'une manière plus agréable que les sorties brutes. Pour utiliser l'ODS, il faut encadrer notre PROC par quelques lignes de code :

```
ods rtf file= « w:\sas\sortiefreq.rtf » ;
 proc univariate data=matable ;
  var mavariable ;
 run ;
ods rtf close ;
```

Ceci crée un fichier sortiefreq.rtf dans le répertoire \\w:\sas\. Ce fichier contient la sortie de la procédure encadrée par l'ODS, ici une proc univariate.
On pourrait souhaiter n'exporter que certains des tableaux édités par la proc. C'est possible !

D'abord on va demander grâce à une instruction **ods trace on ;** à ce que les noms donnés pas SAS à chacun de ces tableaux soit édité dans la LOG. Par exemple :

```
ods trace on ;
 proc univariate data=matable ;
  var mavariable ;
 run ;
ods trace off ;
```

Voilà à quoi ressemble la LOG :

```
1    ods trace on ;
2     proc univariate data=matable ;
3      var mavariable ;
4     run ;


Output Added:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.mavariable.Moments
-------------

Output Added:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.mavariable.BasicMeasures
-------------

Output Added:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.mavariable.TestsForLocation
-------------

Output Added:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.mavariable.Quantiles
-------------

Output Added:
-------------
Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.mavariable.ExtremeObs
-------------


NOTE: PROCEDURE UNIVARIATE used:
      real time           0.57 seconds
      cpu time            0.05 seconds


5    ods trace off ;
```

Admettons qu'on ne veuille exporter que les tableaux des moments et des quantiles. L'instruction ods select *nom_tableau* ; résout le problème :

```
ods rtf file= « w:/sas/sortiefreq.rtf » ;
ods select Moments Quantiles ;
 proc univariate data=matable ;
  var mavariable ;
 run ;
ods rtf close ;
```

Comment sauvegarder quoi ?

Sauvegarder une sortie output :

Save as…
Cette option sauvegarde la sortie sous la forme d'un .lst
Pour ouvrir un .lst, on peut faire File - Open sous SAS, ou bien ouvrir le fichier avec NotePad. On peut aussi insérer le fichier dans un document Word en faisant Insertion - Fichier.

Save as object…
Il s'agit ici de sauvegarder la sortie dans un catalogue SAS. On ouvre le catalogue comme une table, en double cliquant dessus sous SAS. Et dedans on retrouve nos objets. Ces objets sont inaccessibles en dehors de SAS.

ODS
Le Output Delivery System sert à exporter tout ou partie d'une sortie SAS au moment même de sa création. L'exportation se fait en .rtf ou en .html. Les sorties sont plus jolies, et aisément exploitable sous Word. Mais il faut connaître les lignes de code…

Sauvegarder un format :

On peut sauver nos formats dans un catalogue. Au moment de la création du format, on rajoute une option : library=*nom_catalog*.
Par la suite, si on veut utiliser le format, il suffira d'inclure en début de session l'instruction :

option fmtsearch=(*nom_catalog*) ;

Sauvegarder un graphique (qui n'est pas dans l'output) :

Save as image
Il s'agit de sauvegarder le graphique dans un catalogue. On commence par terminer la proc grâce à un quit ; . Puis on clique droit sur le graphique - Save as image. On choisit le catalogue destination et on donne un nom à notre graphique. Le tour est joué.

Option gout=
On peut faire la sauvegarde directement au moment de la création du graphique en utilisant une option : gout=*nom_catalog* ;

Toujours l'ODS.

Pour les graphiques réalisés avec Graph n Go :

Le mieux reste d'exporter le graphique en html par un clic droit - export - html file.
On peut exporter le graphique en html ou en format image avec clic droit - export - html file ou clic droit - export - external file. L'exportation en html enregistre aussi sous format gif.

On peut aussi garder le graphique dans un catalogue en faisant clic droit - export - image entry.

Enfin on peut voir le programme généré par graph n go en faisant un clic droit - export - source file et en cliquant sur preview. On a bien sûr la possibilité d'enregistrer ce programme.

Sauvegarder un résultat insight :

En cliquant sur la flèche en haut à gauche du tableau que l'on souhaite garder, on peut faire save, ce qui exporte le tableau dans la fenêtre output de SAS.

Sinon on peut aller dans le menu File - Save.
Si on a modifié la table et qu'on souhaite enregistrer ces modifications, on choisit Save Data.

Si ce sont les sorties qui nous intéressent, on peut les enregistrer dans un catalogue avec Save Graphics Catalog ou les exporter en format image avec Save Graphics File (tous les tableaux sont alors enregistrés, on peut choisir à ce qu'ils soient dans une même image ou séparés).

Sauvegarder un résultat query :

Une fois que la requête est constituée, on peut faire Show Query. Il apparaît alors une fenêtre avec le code généré par Query pour cette requête.
On a la possibilité d'enregistrer dans un catalogue : sous la forme d'une requête (il suffira de faire clic droit - run) pour la lancer (on choisit alors Save Query - Save as query to include later), ou sous la forme d'un programme SAS (en fait, une PROC SQL) par Save Query - Save as Source entry.
On peut enfin enregistrer la PROC SQL correspondante dans un fichier .sas.
Notez que dans cette fenêtre on peut aussi demander à ce que la requête soit sauvée dans une table (Create table).

PARTIE IV

Introduction aux macros

Dès qu'on s'attaque à des données réelles, il devient courant de devoir répéter le même type de traitement un grand nombre de fois. Par exemple, je dispose d'une table SAS par département, et je dois faire un certain traitement sur ces 96 tables. Bien sûr, je peux toujours soumettre 96 fois le programme, en modifiant le nom de la table à chaque fois… Mais je préfère définir un « macro-programme » qui prendra le nom de ma table en argument !

Un macro-programme n'est autre qu'un programme SAS paramétrable.

Cette partie suit la progression suivante :

D'abord un exemple introductif, qui permet de voir comment l'on passe d'un programme SAS « classique » à un macro-programme.

La définition d'un macro-programme et d'une macro-variable. Comment on déclare et on affecte une macro-variable.

Les macro-fonctions et leur équivalent « procédural », les routines. En particulier, on présente les syntaxes des macro-fonctions et des routines les plus utiles.

Un paragraphe de synthèse qui présente les différents outils de la « charpente » d'un macro-programme : arguments, boucles, etc.

Enfin, un paragraphe sur le stockage et la réutilisation de programmes SAS. On retiendra deux grandes méthodes concurrentes :

- Enregistrement du programme (code) dans un .sas → pour un usage personnel de la macro.
- Enregistrement du programme compilé dans un catalogue SAS → pour partager ses macros avec ses collègues et amis.

Exemple introductif

Je suppose que je dispose d'une table par département, chaque table contenant, pour chacune des communes du département : le code commune, le nombre d'habitants et le nombre d'emplois offerts. Je souhaite sur chacune de ces tables 1) ajouter une colonne txemploi égale au ratio du nombre d'emplois sur le nombre d'habitants 2) connaître la moyenne et l'écart-type de txemploi 3) tracer une boxplot de txemploi, en distinguant deux sous-populations : communes de plus/moins de 10000 habitants.
Pour répondre à ce besoin, je vais écrire une étape data, une proc means, une proc format et une proc boxplot :

```
data dep01 ;
  set dep01 ;
  txemploi = nbemplois / nbhabitants ;
run ;
proc means data=dep01 mean std ;
  var txemploi ;
proc format ;
  value taille low-<10000= « moins de 10000 hab » 10000-high = « plus
de 10000 hab » ;
proc boxplot data=dep01 ;
  format nbhabitants taille. ;
  plot txemploi*nbhabitants ;
run ;
```

Ce programme simple contient 4 occurences du nom de la table. La méthode « à la main » risque d'être puissamment fastidieuse...
Voyons comment je peux transformer mon programme en un **macro-programme** :

```
%macro mapremieremacro (table) ;
  data &table. ;
    set &table. ;
    txemploi = nbemplois / nbhabitants ;
  run ;
  proc means data=&table. mean std ;
    var txemploi ;
  proc format ;
    value taille  low-<10000= « moins de 10000 hab »
                  10000-high = « plus de 10000 hab » ;
  proc boxplot data=&table. ;
    format nbhabitants taille. ;
    plot txemploi*nbhabitants ;
  run ;
%mend ;
```

Les changements intervenus sont minimes :

- Le programme a été encadré par un bloc %macro…%mend permettant la définition du macro-programme.
- Le nom de la table est devenu une **macro-variable** passée en argument de la macro.

Remarque : Le format étant créé une fois pour toute la session, et ne dépendant pas de la table, on peut (doit) le sortir du macro-programme.

Macro-programmes et macro-variables

Syntaxe générale d'un macro-programme

Un macro-programme présente donc la syntaxe suivante :

```
%macro nom_de_la_macro (liste_arguments) ;
   Blocs d'instructions
%mend ;
```

L'appel de la macro s'effectue alors comme suit :

```
%nom_de_la_macro (liste-arguments) ;
```

Appel d'une macro-variable

Les macros utilisent des macro-variables.
Une macro-variable n'est pas inféodée à une table.
Soit x une macro-variable. Le contenu de x est récupéré en écrivant : **&x.**
Le point peut être omis s'il n'y a pas d'ambiguité.

Supposons que l'on ait dix macro-variables mv1, mv2,… mv10 et que l'on veuille effectuer le même traitement sur les 10. On voudrait pouvoir faire une boucle finie, et donc pouvoir appeler les différents mv*i*. Mais dans ce contexte, i est aussi une macro-variable !
Quelles sont les possibilités ?
&mvi. Appelle la macro-variable de nom mvi, qui n'existe pas.
&mv.i Si la macro-variable mv existait, par exemple valant toto, écrire &mv.i reviendrait à écrire totoi.
&mv&i. Le &i. permet bien de récupérer la valeur de i ; mais le &mv renvoie à une macro-variable mv qui n'existe pas.
La solution est : **&&mv&i**.

Les macro-variables contiennent du texte

Une macro-variable est toujours et partout une **chaîne de caractères**.
Si l'on souhaite utiliser sa valeur numérique (à supposer qu'il s'agisse d'une variable à caractère numérique), il faudra forcer l'interprétation en ayant recours à des **macro-fonctions** d'évaluation des nombres.
Parmi ces macro-fonctions, on retiendra :
%eval() qui permet d'interpréter la chaîne de caractères comme un entier.
%sysevalf() qui permet d'interpréter la chaîne de caractères comme un nombre réel.
Ces deux macro-fonctions sont détaillées dans le paragraphe suivant (« les macro-fonctions »).

Déclaration et affectation d'une macro-variable

Les macro-variables sont :

- Soit des arguments du macro-programme
- Soit des variables système
- Soit des variables internes au macro-programme.

Dans ce dernier cas, il existe une instruction réservée à la déclaration et à l'affectation simultanées d'une macro-variable : **%let**. La syntaxe est la suivante :

```
%let ma_macro_variable = mon_texte ;
```

Exemple :
%let toto = je m'appelle toto ;
%let titi = 2 ;

Les macro-variables système

Voici quelques exemples parmi les variables système les plus utiles :

Sysdate	Date et heure courante
Syslast	Nom de la dernière table utilisée
Syserr	Code du type d'erreur rencontré
Syslibrc	0 si le libname s'est bien exécuté
Sysver	Version de SAS utilisée

Les macro-fonctions

Les fonctions SAS usuelles ne sont pas accessibles dans le cadre d'un macro-programme. Il faut avoir recours aux macro-fonctions.

On peut retenir qu'en règle générale, préfixer un mot-clé ou une fonction SAS par le caractère % permet de trouver leur equivalent en macro-langage.

Macro-fonctions de manipulation de chaîne de caractères

%str() Masque les caractères spéciaux
%nrstr() Masque les caractères spéciaux, y compris % et &
%length() Retourne la longueur de l'argument
%index(), **%scan()**, **%substr()** et **%upcase()** fonctionnent comme leurs équivalents en langage SAS usuel (voir dans la partie I les fonctions de manipulation de chaînes de caractères).
%qscan(), **%qsubstr()** et **%qupcase** permettent en outre d'ignorer les caractères réservés % et & (ils sont considérés comme des caractères normaux et donc non interprétés).

Macro-fonctions d'évaluation numérique

%eval() force la chaîne de caractères passée en argument à être interprétée comme une opération arithmétique à résultat entier.

Exemples :
```
%let un = 1 ;
%let deux = 2 ;
%eval (1+2)            Retourne 3
%eval (&un+&deux)      Retourne 3
%eval (&un*2)          Retourne 2
%eval (&un/&deux)      Retourne 0 = partie entière du résultat
```

%sysevalf() force la chaîne de caractères passée en argument à être interprétée comme une opération arithmétique. Un deuxième argument, facultatif, permet de convertir le résultat en entier ou en booléen.

Exemples :
```
%sysevalf (&un/&deux)   Retourne 0,5
%sysevalf(5/2,floor)    Retourne 2 = partie entière de 2,5
%let toto = %sysevalf(&un/&deux)+&un   Affecte à la macro-variable toto le texte « 0,5+1 »
%sysevalf(&toto)        Retourne 1,5 résultat de l'opération
```

La macro-fonction %sysfunc()

Plus généralement, toutes les fonctions SAS peuvent être utilisées dans un macro-programme à condition de passer leur appel en argument de la macro-fonction %sysfunc().

Exemples :
```
%upcase ( « toto » ) est équivalent à %sysfunc ( upcase ( « toto » ) ).
%sysfunc ( int ( ranuni(0) * 100 ) ) tire un entier entre 0 et 100.
```

Les routines

Les routines CALL sont des procédures qui permettent l'interaction entre différentes étapes SAS ou entre différents environnements : lien entre étapes DATA et macro-variables, lien entre commandes DOS et programme SAS, résolution d'une macro-variable, etc.

La routine SYMPUT

Elle permet d'affecter à une macro-variable la valeur d'une variable créée par une étape DATA.
Si la macro-variable n'existe pas, la routine la crée.
La routine est particulièrement utile pour récupérer dans une macro-variable les statistiques calculées par des procédures statistiques.

Exemple :

Dans l'exemple ci-dessous la routine SYMPUT sert à récupérer la moyenne calculée par la PROC MEANS :

```
%macro moyenne (table, var) ;
  proc means data=&table mean ;
    var &var ;
    output out=temporaire mean=moy ;
  run ;
  data _null_ ;
    set temporaire ;
    call symput ( "moyenne", moy ) ;
  run ;
  %put La moyenne de &var est &moyenne ;
%mend ;
```

L'étape DATA ici ne sert qu'à copier la valeur de moy dans la macro-variable moyenne. Elle ne crée aucune table, d'où l'emploi du mot-clé _null_ (voir à ce sujet le dernier paragraphe de la partie I).
Attention à la syntaxe de la routine ! Le premier argument doit être une chaîne de caractère contenant le nom de la macro-variable.

Une méthode équivalente pour réaliser cela est d'avoir recours à une proc sql :

```
%macro moyenne_v2 (table, var) ;
  proc sql ;
    select distinct mean(&var) into :moyenne from &table ;
  quit ;
  %put La moyenne de &var est &moyenne ;
%mend ;
```

On remarque que le programme est plus court, un point de plus en faveur de la proc sql (voir le paragraphe à ce sujet dans la partie IIA).
Ne pas oublier de préfixer le nom de la macro-variable par un « : ».
Note : Ici, l'appel de la routine n'est pas indispensable, puisque la proc sql va déjà imprimer le résultat dans l'output. Mais on peut sans peine imaginer un cas où la macro-variable moyenne servirait à autre chose qu'un simple %put…

La routine EXECUTE

Si l'on a défini une chaîne de caractères contenant une commande SAS, l'appel de la routine EXECUTE sur cette chaîne de caractères permet d'exécuter cette commande.

La routine SYSTEM

Elle permet d'exécuter une commande DOS à partir d'un macro-programme SAS, par exemple la création de fichiers, le déplacement dans l'arborescence des dossiers ou l'appel d'un programme exécutable.

Pour plus de détails, on se réfèrera à l'aide de SAS.

Construire un macro-programme

Les arguments d'un macro-programme

Les macros peuvent prendre un ou plusieurs arguments, précisés entre parenthèses derrière le nom de la macro, et séparés par des virgules. On peut leur faire prendre une valeur par défaut, en écrivant : Mon_argument = ma_valeur_par_défaut.

Exemple :

```
%macro madeuxiememacro (table, stats = mean std) ;
  data &table. ;
    set &table. ;
    txemploi = nbemplois / nbhabitants ;
  run ;
  proc means data=&table. &stats. ;
    var txemploi ;
  proc boxplot data=&table. ;
    format nbhabitants taille. ;
    plot txemploi*nbhabitants ;
  run ;
%mend ;
```

Dans cet exemple, madeuxiememacro prend deux arguments :

- Le nom de la table = macro-variable de nom « table ».
- La liste des statistiques éditées par la proc means = macro-variable de nom « stats » prenant comme valeur par défaut « mean std ».

Règle générale :

Lors de l'appel de la macro :

- On doit renseigner tous les arguments ne prenant aucune valeur par défaut.
- On peut ne pas renseigner les arguments prenant une valeur par défaut ;
- On peut aussi les renseigner, afin qu'ils prennent une autre valeur.

De plus, les arguments sans valeur par défaut :

- Apparaissent toujours en premier dans la liste des arguments.
- Sont « positionnels » : leur place dans la liste des arguments suffit à les définir ie ce n'est pas la peine de rappeler leur nom lors de l'appel de la fonction.

Exemple :

```
%madeuxiememacro (malib.matable) ;
ou :
%madeuxiememacro (malib.matable, stats = sum mean std) ;
```

→ Cette particularité permet d'introduire des paramètres qui servent rarement, sans que l'utilisateur ait à les renseigner à chaque appel de la macro.

Les instructions conditionnelles et itératives

Le corps d'une macro est constitué d'une succession de programmes SAS classiques, à savoir des étapes DATA et des étapes PROC.
Il est possible de faire des boucles comme celles présentées dans le paragraphe « Boucles DO et conditions IF » de la partie I. Les syntaxes sont modifiées comme suit :

```
%DO…%END
    %DO…%TO…%END
    %DO…%WHILE…%END

%IF…%THEN…%ELSE
```

L'instruction %PUT

Elle permet d'afficher un message dans la LOG.

Exemple :
%let toto = 2 ;
%put la valeur de toto est : &toto. ;
→ la LOG affichera le message suivant : la valeur de toto est 2

Stocker et réutiliser un macro-programme

Il est simple d'utiliser un macro-programme. Encore faut-il que le compilateur SAS reconnaisse cette macro ! Plusieurs cas de figure se présentent :

Vous écrivez une petite macro à usage unique ou presque

En ce cas le plus simple est de **compiler la macro à chaque fois** que vous souhaitez l'utiliser. Cette manipulation crée une « macro compilée » qui est placée dans un catalogue de nom « sasmacr » situé dans la WORK (donc perdu à la fin de la session SAS). A chaque appel de la macro, le compilateur va la chercher dans le catalogue en question.

Vous écrivez une macro à usage privé mais récurrent

En ce cas il peut être fastidieux de compiler la macro à chaque session SAS. Il vaut mieux **l'enregistrer dans un .sas** et indiquer au logiciel l'emplacement de ce fichier grâce à une instruction de session option sasautos=.

Plus précisément :
- J'enregistre ma macro macro1512 dans un fichier qui porte le nom macro1512.sas !
- Je soumets (à chaque fois que je veux utiliser macro1512, mais une seule fois par session) l'instruction suivante :

```
option sasautos = (sasautos, « c:\sas\mes_macros ») ;
```

  En supposant que mon fichier macro1512.sas se trouve dans le dossier c:\sas\mes_macros\.

Attention : Si vous modifiez le code de la macro, veillez à la recompiler, car c'est toujours la version présente dans le sasmacr qui prime !

Vous écrivez une macro destinée à être utilisée par d'autres

En ce cas il n'est pas toujours souhaitable de fournir le code source. La solution consiste à créer un **catalogue de macros compilées**, puis à transmettre (ou utiliser soi-même) ce catalogue.

➢ *Pour stocker une macro dans un catalogue :*

Je soumets l'instruction de session suivante :
`option sasmstore = lib` , où lib est le nom de la librairie dans laquelle je veux placer mon catalogue sasmacr de macros compilées.
J'ajoute une option `/stored` à la définition de mes macros et je compile les macros avec cette option.

➢ *Pour utiliser un catalogue de macros compilées :*

Je soumets l'instruction de session suivante :
`option sasmstore = lib` , où lib est le nom de la librairie dans laquelle se trouve le catalogue sasmacr des macros compilées.
J'appelle la macro tout simplement, avec les bons arguments…

Annexes

Afin de faciliter votre recherche dans le polycopié, vous trouverez ci-après :

1. Un index alphabétique des procédures, options, instructions et concepts abordés

2. Une aide thématique du type « Comment faire … ? » classée par grandes catégories :

- Manipulations sur les données
- Statistiques univariées
- Statistiques bivariées
- Modélisations
- Tests
- Graphiques
- Autres

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Comment faire… ?

…Des manipulations sur les données :

MANIPULATION DE TABLES :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Trier une table	PROC SORT	P36
Fusionner deux tables	Instruction MERGE (étape DATA) Ou SAS/SQL : PROC SQL ou outil Query	P20 P39-40 P99
Concaténer deux tables	Instruction SET (étape DATA)	P13
Sélectionner des variables	Instruction KEEP ou DROP (étape DATA)	P13
Sélectionner certaines observations	Instruction WHERE (étape DATA ou PROC) Ou DELETE ou OUTPUT (étape DATA)	P13

MANIPULATION DES ATTRIBUTS DES VARIABLES :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Renommer une variable	Instruction RENAME (étape DATA)	P16
Transformer une variable caractère en variable numérique	Fonction Input ()	P16
Définir la longueur de stockage d'une variable	Instruction LENGTH (étape DATA)	P16
Affecter un label à une variable	Instruction LABEL (étape DATA)	P17
Appliquer un format à une variable	Instruction FORMAT (étape DATA ou PROC)	P17

POUR LE TRAITEMENT DES DONNEES :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Faire des sous-groupes	Instruction BY ou CLASS [1]	P31-48-64
Sélectionner certaines observations	Instruction WHERE [1]	P31
Pondérer les observations	Instruction WEIGHT ou FREQ [1]	P49
Faire du regroupement de modalités	PROC FORMAT pour créer le format, Instruction FORMAT pour l'appliquer [1]	P41
Attribuer un rang à une modalité	PROC RANK	P42-43
Désigner une variable comme identificateur	Instruction ID [1]	P66

…Des statistiques univariées :

SUR VARIABLE QUALITATIVE :	PROC FREQ	PP45 à 47
Nombre d'occurrences d'une modalité **Pourcentage d'occurrence d'une modalité**		

[1] Dans les procédures concernées

SUR VARIABLE QUANTITATIVE **:**

Toutes les statistiques simples recensées ci-dessous figurent dans la sortie standard de la PROC UNIVARIATE.

Ce que je veux calculer	*Option de la PROC MEANS adéquate*
Nombre d'observations non manquante	N
Nombre de valeurs manquantes	NMISS
Somme	SUM
Moyenne	MEAN
Ecart-type	STD
Variance	VAR
Minimum	MIN
Maximum	MAX
Coefficient d'asymétrie	SKEWNESS
Coefficient d'aplatissement	KURTOSIS
Somme des carrés	USS
Somme des carrés des écarts à la moyenne	CSS
Médiane	MEDIAN
Mode	MODE
Distance interquartile	QRANGE
Quantiles	Q1 (premier quartile) – Q3 (troisième quartile) P10 (premier décile) – P90 (neuvième décile) P1 (premier centile) – P99 (dernier centile) – P5 – P95 Pour les autres centiles, PROC UNIVARIATE.

…Des statistiques bivariées :

STATISTIQUES BIVARIEES SUR VARIABLE QUALITATIVE **:**

Ce que je veux éditer	*Comment SAS y répond*
Occurrences d'un profil de modalités	PROC FREQ
Distance du Chi-deux	PROC FREQ option chisq
Statistiques dérivées du Chi-deux	PROC FREQ option chisq
Contributions à la distance du Chi-deux	PROC FREQ option cellchi2

[1] Dans les procédures concernées

STATISTIQUES BIVARIEES SUR VARIABLE QUALITATIVE ORDINALE **:**

Ce que je veux éditer	*Comment SAS y répond*
Tau-b de Kendall	PROC CORR option kendall
Corrélation des rangs de Spearman	PROC CORR option spearman

STATISTIQUES BIVARIEES SUR VARIABLE QUANTITATIVE **:**

Ce que je veux éditer	*Comment SAS y répond*
Coefficients de corrélation (Pearson)	PROC CORR ou Option CORR [1]
Matrice des covariances	PROC CORR option cov
Coefficients de corrélation partielle	PROC CORR option partial
Mesure de liaison (D) de Hoeffding	PROC CORR option hoeffding

…Des modélisations :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Spécifier un modèle	Instruction MODEL [1]	P66
Imposer une restriction à un modèle	Instruction RESTRICT [1]	P67
Effectuer une sélection de variables dans un modèle	Option SELECTION= [1]	P70
Détecter les individus atypiques d'un modèle	Option R [1]	P67
Détecter les individus influents d'un modèle	Option INFLUENCE [1]	P67
Effectuer une régression linéaire multiple	PROC REG	PP66 à 78
Effectuer une régression sur variable catégorielle	PROC LOGISTIC	PP79 à 82
Effectuer une modélisation à plusieurs équations	PROC SYSLIN ou PROC MODEL	P83-84

…Des tests :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Test d'adéquation à une loi normale	PROC UNIVARIATE option normal	P53
Test de nullité de la moyenne	PROC UNIVARIATE	P53
Test d'égalité des moyennes	PROC TTEST	P64-65
Choix du seuil pour le calcul d'intervalles de confiance	Option ALPHA [1]	P66
Test de Durbin Watson	Option DW [1]	P67
Test de White	Option SPEC [1]	P67
Test de liaison entre paramètres estimés d'un modèle	Instruction TEST [1]	P67

[1] Dans les procédures concernées

[1] Dans les procédures concernées

…Des graphiques :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Boîte à moustaches	PROC UNIVARIATE option PLOTS Ou PROC BOXPLOT	P53 P91
Diagramme stem and leaf	PROC UNIVARIATE option PLOTS	P53
Diagramme d'ajustement à une loi normale	PROC UNIVARIATE option PLOTS	P53
Graphe des résidus d'une modélisation	Instruction PLOT [1]	P66
Histogrammes	PROC GCHART instruction HBAR ou VBAR	P84
Diagrammes circulaires	PROC GCHART instruction PIE	P84
Diagramme en bâtons ou circulaires dont la hauteur (resp. les angles) varie avec la valeur d'une variable quantitative	Option SUMVAR= de la PROC GCHART	P85
Nuage de points de y en fonction de x	PROC GPLOT	P87
Graphe (relié) y en fonction de x	PROC GPLOT avec instruction SYMBOL INTERPOL=	P87
Graphes superposés : y et z en fonction de x	Option OVERLAY ou instruction PLOT2 de la PROC GPLOT	P87-88

…Autre chose :

Ce que je veux faire	*Comment SAS y répond*	*page indicative*
Allouer une librairie	Instruction LIBNAME	P6
Importer des données	Module SAS/Wizard ou PROC IMPORT	P33
Lier un fichier de données externes	Instruction FILENAME	P33
Voir les propriétés d'une table	PROC CONTENTS ou Clic droit – Properties	P34-35
Faire des modifications « à la main »	SAS/INSIGHT ou Menu Edit - Edit Mode	
Récupérer les résultats dans une table	Instruction OUTPUT	PP31-49-54
Enregistrer un résultat SAS (autre)	Sauvegarde dans un catalogue ou un fichier	P102
Exporter une sortie SAS	Output Delivery System	P100
Utiliser l'aide en ligne de SAS	Menu Help – SAS System Help	P94
Vider le contenu d'une fenêtre	Commande Clear	P8
Imprimer une table dans l'OUTPUT	PROC PRINT	P36