Un peu de SAS dans un grand verre...

Gilles Hunault

Université d'Angers, 2001

Table des matières

Préface

Chapitre 1.

Présentation de SAS

1.1 Qu'est-ce que SAS ?

SAS est à la fois un système de gestion de fichiers, un outil de calcul statistiques, un atelier de développement de programmes multi-plateformes (dos, unix, mac mais aussi vm/sp, mvs etc.). Pour qu'on puisse efficacement communiquer avec *SAS*, les développeurs de SAS ont créé un langage de commandes et de programmation, le **SASL** (ou "langage de *SAS*").

Nous présentons ici les interfaces de *SAS*, le langage de *SAS* et comment réaliser des calcus de Statistique Descriptive en *SAS*. Nous laissons le soin au lecteur, à la lectrice de vérifier tout ce que nous écrivons en effectuant les travaux pratiques correspondant sur ordinateur.

On trouvera aussi en fin de document quelques remarques sur ce qu'est aussi *SAS*, à savoir un logiciel "monstre", qui fait "tout" et qui peut "tout" ou presque – mais à condition d'y mettre le prix et d'investir dans la programmation *SAS*. Par exemple, quel logiciel est utilisé par la Banque de France depuis 30 ans pour gérer ses fichiers et ses données ? Certainement pas *Microsoft Excel*, mais peut-être *SAS*...

1.2 Les trois interfaces de SAS

On communique avec *SAS* au moyens d'instructions qui se termine obligatoirement par un point-virgule. Les interfaces de base sont respectivement les fichiers, le clavier et la souris.

1.2.1 Interface Ligne

Les instructions *SAS* peuvent être écrites dans un fichier et soumises en mode ligne à Sas. L'interface est alors minimale : *SAS* se contente d'exécuter le fichier, disons CALC.SAS et met dans les deux fichiers CALC.LOG et CALC.LST le résultat de l'exécution. CALC.LST contient les calculs effectués ; CALC.LOG, quant à lui, décrit ce que *SAS* a fait, ce qu'il a compris, combien de temps a duré chaque étape...

1.2.2 Interface fenêtrée

L'interface traditionnelle se compose de trois fenêtres : la fenêtre *Program editor* ou plus simplement *Editor* (accessible par *F5* ou en activant la commande *Program Editor* du menu *Fenêtre*), la fenêtre *Log* (*F6*) et la fenêtre *Output* (*F7*). On écrit les instructions dans la fenêtre Editor et on soumet (par *F3* ou par la commande Exécuter du menu *Général*) ces instructions à *SAS*. On visualise alors les résultats dans la fenêtre *Output* et en cas de problème on vient lire le détail des actions effectuées ou ratées par *SAS* dans la fenêtre *Log*. Une fois soumis, le texte dans Editor disparait mais on peut le rappeler par la touche *F4* (ou la commande *Rappeler le texte* du menu *Local*).

1.2.3 Interface assistée

L'interface d'assistance de *SAS* se lance par la commande *Sas/Assist* du menu *Général*. On dispose alors d'icones et de panneaux à cliquer qui réduisent au maximum le temps de frappe (mais masquent les instructions). On exécute alors le calcul par la commande *Exécuter* du menu *Local*. Il est possible de sauvegarder le texte des instructions correspondant aux "clics" en activant la commande *Visualiser le code source* du menu *Local*.

On sort du mode *Assist* en activant les différentes commandes *Retour* jusqu'au moment où on arrive au panneau principal de *Sas/Assist* dont on sort en activant la commande *Sortie*. Cette interface est utile pour débuter, pour se faire une idée de l'ensemble des modules. Par exemple, si dans le *Panneau Principal* on active le panneau *Environnement* puis la ligne *Tables SAS d'essai*, *SAS* propose de créer (ou de recréer) des exemples de fichiers de données de façon à pouvoir tester des calculs sur des fichiers ad hoc.

Au lieu de taper des instructions dans la fenêtre *Editor*, on peut également faire du copier/coller à partir d'un autre éditeur, ou charger un ensemble d'instructions par la commande *Open* du menu *File*, ou encore, et c'est la solution que nous préférons, inclure le texte du programme par l'instruction

```
%include 'nom_du_fichier'.sas ;
```

De même, nous conseillons d'exécuter la commande *Nouveau* dans la fenêtre *Output* avant de relancer une suite de commandes de façon à être bien sûr(e) de visualiser les résultats des commandes et non pas ceux de la série précédente.

Lors de l'inclusion de fichiers, il est conseillé de mettre l'option globale Source2 à On pour les lignes incluses avec leur numéro apparaissent dans le *Log*.

1.3 Comment apprendre SAS ?

Tout d'abord en s'armant de patience !

SAS est un système logiciel complet, avec une mécanique très lourde mais efficace. Le lanagage de sas, qui se veut langage exhaustif de traitement de données, permet de tout faire, à condition d'utiliser la bonne procédure (ou plutot le bon paramétrage de la bonne procédure).

Il faut en général du temps pour se familiariser avec la philosophie de *SAS*, sa façon de parcourir les données, sa gestion ligne par ligne des variables.

Par exemple, pour cumuler des ventes, on pourrait croire que le programme

```
data ventes ;
input ventes ;
if _n_ = 1 then cumul = ventes ;
           else cumul = cumul + ventes ;
datalines ;
100
300
200
;
proc print data=ventes ;
run ;
```

est suffisant. C'est une erreur : la variable cumul "disparait" d'une ligne de données à l'autre et l'affichage des résultats à savoir :

```
OBS    VENTES    CUMUL

 1       100      100
 2       300        .
 3       200        .
```

semble donc incompréhensible au profane...

Pour apprendre SAS, il faut commencer par connaitre le nom des procédures de base, puis lire leur syntaxe complète, la liste de leurs options. Cela demande de nombreux essais (parfois un essai pour chaque option), avec un apprentissage parfois des difficultés conceptuelles. Ainsi keep ne retient que les variables citées, drop ne retient pas que les variables passées en paramètres. Et si on effectue[1] un keep et un drop de la même variable ? ou si on "droppe" tout, que fait SAS ? Essayer sur un exemple n'est parfois pas suffisant et il faut tout le $feeling$ du gourou es SAS pour parfois trouver la solution.

[1] seul un enseignant se poserait la question aussi directement ; en fait, par le jeu de substitution de variables, il peut arriver de faire un keep et un drop d'une même variable...

Chapitre 2.

SAS, langage de commandes

Le langage de *SAS* est à la fois un langage de commandes et de programmation. Commençons par les commandes. Il y en a trois fondamentales : DATA, PROC et RUN. DATA permet de décrire les données et les fichiers associés, PROC définit la procédure de calcul à utiliser, RUN ne sert qu'à exécuter le traitement. Elle est obligatoire en fin de texte avant exécution. L'instruction DATA n'est pas obligatoire. Par exemple, si les tables d'essais de *SAS* ont été créées comme indiqué au paragraphe précédent, les deux lignes

```
* tut01.sas ;
proc print data=sasuser.habitat ; run ;
```

affichent le contenu du fichier d'exemple nommé habitat.

2.1 L'instruction DATA

Le mot DATA indique en fait le début de toute une "étape des données". Cette étape ne s'arrête que lorsqu'on rencontre l'instruction PROC ou une autre instruction DATA ou l'instruction RUN. Pour de petits jeux de données ou des calculs "jetables", on utilise une étape DATA/INPUT/DATALINES qui se prépare comme suit. On met après DATA le nom du fichier à créer sous forme d'un seul mot d'au plus 8 caractères. On met ensuite un point-virgule pour indiquer que la ligne DATA est terminée et on décrit par la ligne INPUT le format des données (voir exemple). Après l'instruction DATALINES terminée elle-aussi par un point-virgule, on met les données à utiliser à raison d'une ligne de données par ligne physique et enfin on termine l'étape par un dernier point-virgule.

Par exemple, pour jouer à la marchande, nous voulons comptabiliser des ventes. On dispose du prix unitaire PU et du nombre d'articles NA. On veut simuler les deux ventes NA=10, PU=8 et NA=3, PU=11. On écrira pour la partie définition des données :

```
* tut02.sas ;
data marchan ;
 input na pu ;
datalines ;
10 8
3 11
;
run ;
```

Lorsqu'on dispose de données plus conséquentes, il vaut mieux laisser les données dans un fichier externe, les référencer par l'instruction FILENAME et le charger par l'instruction INFILE. Si les donnés ne sont pas seulement délimitées (comme dans l'exemple précédent) mais bien cadrées, dizaine sous dizaine, unité sous unité, on donne à *SAS* les positions physiques des variables dans la ligne des données à la suite du nom de la colonne. Ainsi avec le fichier-programme

```
* tut03.sas ;
filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data essai ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

** pour vérification ;

proc print data=essai ;
run ;
```

SAS comprend que nos données sont dans le fichier z :\elf.dat, que les quatre premiers caractères de la ligne forment la colonne ID et que cette colonne (à cause du symbole $) doit être traitée comme du texte et pas comme du numérique, que la colonne SEXE correspond au caractère numéro 9 dans la ligne etc.

Les fichiers créés sont temporaires et n'existent pas lorsqu'on termine la session *SAS*. Il est possible de créer des fichiers dits permanents mais nous ne le détaillons pas ici, pas plus que le vocabulaire propre à *SAS* (librairie, référence, comme avec libname, libref, fileref, filename...).

Dans l'étape DATA, on peut également donner un libellé (LABEL) à une colonne, lui associer un format de description (mais il faut définir ces formats par la procédure FORMAT). FORMAT définit de nouvelles variables (en général texte) qui font la correspondance entre une valeur (ou une plage de valeurs, comme 12-20 pour indiquer entre 12 et 20, bornes comprises) et une chaine de caractères. Attention : il faut mettre un point après la variable de FORMAT dans l'option FORMAT de DATA car plusieurs variables peuvent partager le même format et *SAS* a besoin du point pour s'y retrouver...).

Voici un exemple de telles opérations :

```
* tut04.sas ;
proc format ;
 value Sexf 0='Homme' 1='Femme' ;

data work.elf ;
 infile 'l:\gh\crs\stat\trysas\elf.dat' ;
 input ID $ 1-4 SX 9 ;
 label SX='Sexe de la personne' ;
 format SX Sexf. ;

** pour vérification ;

proc print data=work.elf ;
run ;
```

Enfin, dans l'étape DATA, on peut changer de nom base avec SET, recoder les données ou construire de nouvelles colonnes, détruire des lignes via DELETE. Par exemple, pour la marchande, nous voulons calculer les sommes gagnées pour les plus grosses ventes. On décide pour cela de calculer le prix de vente PV égal au produit du nombre d'articles par le prix unitaire et de ne pas retenir les lignes telles que pu < 5 ou pv < 30. On veut nommer VENTES la table *SAS* correspondante.

Voici les instructions.

```
* tut05.sas ;
data work.marchan ;
 input na pu ;
 pv  = na*pu ;
 if pu <  5 then delete ;
 datalines ;
 10 8
 3 11
 1  1
 7  2
 1  7
 10 50
 50 12.5
;

data work.ventes  ;
 set work.marchan ;
 if pv < 30 then delete ;

** pour vérification ;

proc print data=marchan ;
run ;

proc print data=ventes ;
run ;
```

2.2 L'instruction PROC

Le mot PROC indique le début d'une étape-procédure. Le mot suivant est le nom de la procédure, suivi du mot DATA= pour indiquer quelles sont les donnés. Suivent alors les paramètres éventuels de la procédure avec éventuellement un / pour spécifier des options. Les autres options de l'étape procédure sont définies par des mots propres à la procédure, par exemple VAR pour spécifier la liste (et surtout l'ordre) des variables à utiliser, BY pour afficher de faççon groupée. Les procédures usuelles sont PRINT, MEANS, CORR, FREQ, PLOT, CHART.

Elles permettent respectivement d'imprimer, d'effectuer des calculs statistiques de base sur des données quantitatives et des comptages, de tracer des courbes et histogrammes.

La première procédure à savoir utiliser est certainement PRINT. Elle affiche les données. Ainsi, après l'étape DATA, on met souvent (au moins en début de session *SAS*) une instruction PROC PRINT pour vérifier que les données ont été correctement lues. Une étoile en début de ligne indique un commentaire, c'est à dire une ligne ignorée par *SAS*. Par exemple :

```
* tut06.sas ;
* création du jeu d'essai pour la marchande ;
data marchan ;
 input na pu ;
datalines ;
10 8
3 11
;
* affichage des données ;
proc print data=marchan ;
run ;
```

Pour imprimer de façon ordonnée, il faut utiliser le mot BY après avoir trié via la procédure SORT les données. Ainsi dans le dossier Elf, nous voulons utiliser l'affichage de politesse, à savoir les femmes d'abord, les hommes ensuite, les personnes d'age élevé avant les plus jeunes. Décidant de n'afficher que l'identificateur, le sexe et l'age des personnes dans cet ordre, nous écrirons les lignes suivantes

```
* tut07.sas ;

* définition des données ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

run ;
```

```
/* tri selon l'ordre de politesse ;
   femme d'abord, homme ensuite
   les vieux d'abord, les jeunes ensuite
   code sexe : 0=homme, 1=femme           */

proc sort data=elf  out=elf_Poli;
  by descending sexe descending age ;

* affichage sélectif dans l'ordre ;

proc print data=elf_Poli ;
  var id sexe age ;

* exécution ;

run ;
```

2.3 Quelques procédures utiles

La procédure CONTENTS permet de connaitre ce que contient une table sas. Exemple d'utilisation :

```
* tut08.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

* vérification du stockage de données ;
proc contents data=marchan ;

proc contents data=sasuser.habitat ;

proc contents data=elf ;

run ;
```

En particulier, pour savoir ce que contient tout la librairie utilisateur :

```
/*
   +================================================+
   +                                                +
   +  attention : cela produit des tonnes de pages, +
   +  mais au moins on a tout le détail !           +
   +                                                +
   +================================================+

*/

proc contents data=sasuser._all_ ;
run ;
/*
*
* au mieu de sasuser, on peut mettre aussi
* library, maps, sashelp, work
*
*/
```

2.4 Fichiers temporaires et permanents

Une référence de fichier (*fileref*) au sens de *SAS* est le nom attribué aux données en cours de traitement. Si ce nom se compose d'un seul mot, comme par exemple ventes alors le fichier est temporaire et disparaitra lorsqu'on quitte *SAS*. Si ce nom est composé de deux mots reliés par un point, alors le fichier est permanent. Le premier mot fait référence à une "librairie" (*libref*) et le second mot est le nom du fichier. Le fichier sera écrit à l'endroit correspondant pendant la session. On associe un nom de répertoire à une libraire par la commande libname.

Supposons par exemple qu'on exécute les commandes suivantes dans une session *SAS* :

```
libname enquet 't:\stat\enq\99\surv' ;
data testUn ;
...
data cours.testDeux ;
...
```

alors le *dataset* référencé par testUn est temporaire alors que alors le *dataset* référencé par enquet.testDeux est permanent et est stocké dans le fichier t :\stat\enq\99\surv\testDeux.sd2.

A l'usage, ce mécanisme de référence se révéle souple : d'un ordinateur à l'autre, il suffit de modifier le *libname* pour que tous les fichiers correspondent. Sur un même ordinateur, changer de *libname* permet d'utiliser des fichiers de même nom situés dans des répertoires différents.

Chapitre 3.

Calculs statistiques en SAS

3.1 Traitement des Variables Quantitatives

La procédure de base est MEANS. *SAS* l'appique à toutes les colonnes numériques (même si les nombres correspondent à des codes). Par défaut, *SAS* affiche le nombre de valeurs (N), la moyenne (MEAN), l'écart-type (STD), le minimum (MIN) et le maximum (MAX). Ainsi pour notre dossier Vins qui ne contient que des variables numériques quantitatives, il suffit d'écrire

```
* tut10.sas ;

filename vins 'l:\gh\crs\stat\trysas\vins.dat' ;
data vins ;
 infile vins ;
 input TYPEV $ BELGIQ NEDERLD RFA    ITALIE
               UK     SUISSE  USA    CANADA ;
 list ;

proc means data=vins ;
run ;
```

Pour n'afficher que certaines caractéristiques, on les indique après la spécification de données. Si on veut certaines variables seulement, on utilise VAR et si on veut traiter des sous-populations, on écrit BY (il est nécessaire d'avoir trié auparavant).

Ainsi pour avoir le nombre de personnes, la moyenne et le coefficient de variation (CV=σ/m) de l'age pour chaque sexe du dossier Elf, on écrit

```
* tut11.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

proc sort data=elf out=elf_Tri ;
 by sexe ;

proc means data=elf_Tri n mean cv maxdec=1;
 var age ;

proc means data=elf_Tri n mean cv maxdec=1;
 var age ;
 by  sexe ;

run ;
```

Il y a d'autres paramètres disponibles pour MEANS dont NMISS, nombre de valeurs manquantes, MAXDEC pour préciser le nombre de décimales à l'affichage, RANGE pour l'étendue (c'est la différence maximum moins minimum), SUM pour la somme des valeurs, VAR pour la variance, SKEWNESS, KURTOSIS, SUMWGT, CSS, STDERR... La seconde procédure à utiliser est CORR. Elle calcule les divers coefficients de corrélation (le ρ classique de Pearson, mais aussi celui de Spearman et celui de Kendall pour les rangs, les coefficients de Cronbach, de Hoeffding etc.). Par défaut, *SAS* calcule les caractéristiques m, σ etc. de toutes les variables numériques puis affiche une matrice de corrélation avec en-dessous de ρ, la probabilité de dépasser $|R|$ dans l'hypothèse H_0 (hum !) et le nombre d'observations sur lesquelles ρ a été calculé.

On peut bien sur avec CORR spécifier des corrélations partielles, faire des corrélations entre une variable et une liste d'autres variables, etc., n'afficher que les n meilleures corrélations (avec BEST=n mais attention la machine donne toujours $\rho(X,X)=1$ donc BEST doit être toujours supérieur à 2) etc.

Pour obtenir la matrice des corrélations classiques, on prend les options NOPROB et PEARSON, soit, pour le dossiers Vins :

```
* tut12.sas ;

filename vins 'l:\gh\crs\stat\trysas\vins.dat' ;
data vins ;
 infile vins ;
 input TYPEV $ BELGIQ NEDERLD RFA    ITALIE
               UK     SUISSE  USA    CANADA ;

proc corr data=vins pearson noprob ;

run ;
```

Le complément de MEANS est UNIVARIATE, celui de CORR est REG. Nous renvoyons le lecteur, la lectrice au manuel de référence, à l'aide "en ligne" pour en lire les détails et les options, fort nombreuses...

3.2 Traitement des Variables Qualitatives

La procédure de base est ici FREQ. Avec l'option TABLE, elle calcule les tris à plat en indiquant au passage les effectifs absolus et relatifs, le cumul des fréquences. Par exemple :

```
* tut13.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

proc freq data=work.elf  ;
  tables sexe etud prof ;
run ;
```

On peut bien sur choisir de ne pas afficher le cumul avec NOCUM, afficher par ordre décroissant d'effectifs avec ORDER=FREQ.

Par exemple on peut écrire :

```
* tut14.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

proc freq data=elf order=freq ;
  tables sexe etud prof / nocum ;

run ;
```

Pour un tri croisé, on écrit la même chose, sauf que dans la ligne TABLES on donne les couples de variables à croiser reliés par une étoile.

Ainsi pour le tri croisé sexe / études dans le dossier Elf, on écrira :

```
* tut15.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

proc freq data=elf  ;
  table sexe*etud ;

run ;
```

Pour obtenir un tri croisé "bien beau, bien propre", on utilisera des labels et des formats comme suit :

```
* tut16.sas ;

proc format ;
 value sexf  0='Homme' 1='Femme' ;
 value etudf 0='NR' 1='Primaire' 2='Bepc' 3='Bac' 4='Sup' ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data   elf_beau ; infile elf ;
```

```
input  ID $ 1-4 SX 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;
label  SX='Sexe de la personne' ;
label  ETUD='Niveau d''études' ;
format SX sexf. ETUD etudf. ;

proc freq data=elf_beau ;
 table etud*sx / norow nocol nopercent ;

run ;
```

Signalons que la procédure TABLES calcule aussi des chi-deux, les coefficients de Fisher, de Stuart, Somer, Cramer, Mantel-Haenszel...

3.3 Tracé de courbes

On trace avec la procédure PLOT. L'instruction PLOT (si, si, c'est bien le même mot) est alors l'option de tracé. La syntaxe du choix des variables est, comme pour FREQ, y*x qui vient tracer y en fonction de x. Par exemple :

```
* tut17.sas ;

data ventes ;
input an vente ;
datalines ;
1995 127000
1996 150000
1997 134000
1998 186000
;
proc plot data=ventes ;
 plot vente*an ;
run ;
```

SAS met alors un A comme symbole de tracé, un B si deux points sont superposés sur le graphique. On peut choisir son symbole (ou faire mieux, si on n'imprime pas dans un fichier mais si on utilise un "grpahique écran"...). Avec les options hpercent et vpercent, on peut même plusieurs graphiques sur une même page.

3.4 Tracé d'histogrammes

Les histogrammes se tracent avec la procédure CHART, que ce soit en barres (ou histogrammes verticaux, option VBAR) ou en lignes, (ou histogrammes horizontaux, option HBAR) ou en secteurs angulaires (option PIE). *SAS* se charge de calculer les fréquences.

Si ces histogrammes sont faciles à produire, on se méfiera du fait qu'il sont en standard en mode texte, c'est à dire que ce sont des lignes de caractères "normaux" et non pas des images, comme avec *Excel* ou *Gnuplot*.

Pour l'histogramme des niveaux d'étude dans le dossier Elf, on écrira :

```
* tut18.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

proc chart data=elf ;  vbar sexe ;

proc chart data=elf ;  hbar sexe ;

run ;
```

La ventilation d'un histogramme se fait avec l'option GROUP. Signalons qu'on peut réaliser des graphiques en trois dimensions avec BLOCK plutot que HBAR, VBAR ou PIE.

```
* tut19.sas ;

filename elf 'l:\gh\crs\stat\trysas\elf.dat' ;
data elf ;
 infile elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18
       ETUD 23 REGI 28 ACTIO 33 ;

proc chart data=elf ;  vbar sexe / group=etud ;

run ;
```

Finissons maintenant par un exemple plus conséquent. Reprenant le dossier Vins, nous décidons d'afficher les résultats statistiques par moyenne décroissante puis par ordre de coefficient de variation décroissant. Pour cela, nous stockons avec OUT les résultats de la procédure MEANS sans rien afficher (NOPRINT). Le fichier de résultat ne se présentant pas sous une forme convenable pour notre tri, nous le transposons avec la procédure TRANSPOSE. Ensuite, nous renommons les colonnes avec un identifiant plus explicite (*moy* plutot que *col*4, etc.). et nous calculons à la main le *cdv*. Il ne reste plus qu'à trier par ordre décroissant (SORT... BY DESCENDING...) et à afficher pour obtenir ce que l'on veut ! Voici donc le programme complet :

```
* tut20.sas ;

** 1. Lecture des données ;

filename vins 'l:\gh\crs\stat\trysas\vins.dat' ;
data work.vins ;
 infile vins ;
 input TYPEV $ BELGIQ NEDERLD RFA    ITALIE
               UK     SUISSE  USA    CANADA ;

** 2. Calculs ;

proc means  data=work.vins  noprint ;
     output out=work.vins_gsa ;

** 3. Transposition du fichier de résultats ;

proc transpose data=work.vins_gsa out=work.vins_asg;

** 4. Suppression des lignes "parasites"
      et calcul du cdv ;

data work.vinsasg ;
     set work.vins_asg ;
     if col1=0 then delete ;
     if _name_='_FREQ_' then delete ;
     min = col2 ; max = col3 ;
     moy = col4 ; ect = col5 ;
     cdv =100.0*ect/moy ;
```

```
** 5. Tri par moyenne décroissante ;

proc sort  data=work.vinsasg ;
 by descending moy ;

proc print data=work.vinsasg ;
 var _name_ moy ect cdv min max ;

** 6. Tri par coefficient de variation décroissant ;

proc sort  data=work.vinsasg ;
 by descending cdv ;

proc print data=work.vinsasg ;
 var _name_ moy ect cdv min max ;

** Exécution ;

run ;
```

Et dans la foulée, faisons l'AFC de ce même tableau :

```
proc corresp   data=work.vins  ;
     var BELGIQ NEDERLD RFA    ITALIE
         UK     SUISSE  USA    CANADA ;
run ;
```

vous voulez plus simple ? Y'a pas !

3.5 Traitement des Variables Textuelles

3.6 Autres calculs

Chapitre 4.

SAS, programmation avancée

4.1 Bonjour chez vous avec la table !

Prenons notre fameux exemple traditionnel de "bonjour" qui demande un nom, affiche la date et l'heure et qui dit au revoir à la personne en traduisant le nom en majuscule. La version minimale qui convertit le nom en majuscule peut être

```
* bonjour1.sas ;

data bonjour ;
     input   nom  $   ;
     nomMaj  = upcase(nom) ;
     drop    nom ;
     datalines             ;
paul
;
```

mais pour inclure la date et l'heure, c'est un peu plus délicat car il faut utiliser des fonctions comme *day* et *date* et concaténer les chaines de caractères avec l'opérateur ||.

De plus, pour qu'il n'y ait rien d'autre sur la page, il faut mettre un titre vide avec title ' '; et activer l'option nodate. Notre programme crée des variables autres que celle définie dans la partie input de l'instruction data et on ne garde que la variable intéressante avec l'instruction keep.

D'où la deuxième version de bonjour.sas :

```
* bonjour2.sas ;

title ' ' ;
options nodate ;

data bonjour ;
 input   nom  $   ;
 lejour    = day(date()) || '/'  ;
 lemois    = month(date()) || '/' ;
 lannee    = year(date()) || ',' ;
 dateheur  = 'Le ' ||
             compress(lejour || lemois || lannee) ||
             ' au revoir, ' ;
 msg       = compbl(dateheur||compress(upcase(nom)|| '.')) ;
 keep msg ;
 * put écrit dans le log  ;
 put  msg ;
 datalines ;
paul
;
```

Passons maintenant à la traditionnelle table de multiplication. Un premier essai donne :

```
* tablemul1.sas ;

data tablemul ;
 input n ;
 do i = 1 to n ;
    msg1 = ' fois ' ;
    l    = n ;
    msg2 = ' = ' ;
    k    = i*n ;
    output ;
 end ;
 drop n i ;
 datalines ;
6
;
```

Au lieu de mettre les données dans la partie datalines de l'instruction data, il est possible de lire les données d'un fichier externe (nommé ici t :table.mul) grâce à la commande infile.

```
* tablemul2.sas ;

options nodate nocenter nonumber linesize=256 pagesize=32767 ;
filename tablemul 't:\table.mul' ;

     data tablemul ;
      infile tablemul ;
      input n ;
      title ' Table de Multiplication de ' n  ;
      do i = 1 to 10 ;
         msg1 = ' fois ' ;
         l    = n ;
         msg2 = ' = ' ;
         k    = i*n ;
         output ;
      end ;
      drop n i ;
```

Il peut être parfois intéressant de mettres les résultats dans le *Log* avec l'instruction put, de format les résultats... C'est ce que nous faisons avec le programme suivant :

```
     * tablemul3.sas ;

     data tablemul ;
       nombre = 10 ;
       fois   = ' fois ' ;
       egale  = ' = ' ;
       do i = 1 to 10 ;
          j = i * nombre ;

          * résultats dans le log ;
          put i 2. fois $ 6. nombre egale j 7. ;

       end ;
     run ;
```

4.2 La classe, cumuls et variables automatiques

Finalement, le problème "classe" qui calcule les moyennes des élèves est assez simple, car SAS est "étudié pour".

```
* classe.sas ;
filename classe 't:\classe.dat' ;
data classe ; infile classe ;
   input  nom $  not1 not2 not3 ;
   moy =(not1+not2+not3)/3 ;
   put nom $ 1-40 not1 not2 not3 moy ;
proc print ; run ;
```

On notera qu'en sortie on force la taille de nom à 40 pour obtenir un "bon" cadrage. Dans le même genre d'idées, si l'on connait l'instruction retain, rien de plus simple que de cumuler le total des ventes dans la colonne cumul, à l'aide de la variable automatique _n_ qui correspond au numéro d'enregistrement, que l'on n'affiche pas dans la procédure print grâce à l'option noobs.

Nous en profitons au passage pour montrer l'utilisation de

data a ; set b ;

qui transfère dans l'ensemble de données (*dataset* dans la terminologie *SAS*) nommé a les données de b.

```
 data ventes ; input fruit $ nbkilo prixunit ;
   datalines ;
   oranges  3  5.5
   pommes  10  2
   oranges  5  5.5
   ;
data ventcum ; set ventes ;
   vente  = _n_ ;
   retain cumul ;
   if _n_ = 1
      then cumul = nbkilo*prixunit ;
      else cumul = cumul + nbkilo*prixunit ;
   keep vente cumul ;
proc print data=ventcum noobs ; run ;
```

4.3 Combien de lignes, combien de colonnes...

Pour connaitre le nombre de lignes, de colonnes d'un fichier, rien ne vaut quelques bonnes fonctions de derrière les fagots ! Ce sera l'occasion ici d'introduire le langage des macros de *SAS*.

SAS dispose d'un deuxième langage de programmation, complémentaires du premier, dont les structures ne sont plus nommées programmes mais macros-commandes. Ce deuxième langage, plus souple, plus interprété, utilise des variables repérées par &, des mots-clés repérés par %. Une macro commence par le mot clé macro, se termine par le mot-clé mend (contraction de *macro end*). Elle s'utilise par %nom_macro(paramètres). Par exemple, pour utiliser la macro suivante nommée combien sur le *dataset* référencé par cours.vins, il suffit d'écrire

%combien(cours.vins) ;

après avoir éventuellement chargé la macro par

%include 'combien.sas' ;

On appréciera le fait qu'il n'y ait pas à écire run ; ici. Le résultat (la phrase dans le fichier...) est mis dans le *Log*.

```
/*******************************
 *                             *
 * macro SAS : combien         *
 *                             *
 *******************************/

    %macro combien(ds) ;
    %let dset  = &ds ;
    %let dsid  = %sysfunc(open(&dset)) ;
    %let nblig = %sysfunc(attrn(&dsid,NOBS)) ;
    %let nbcol = %sysfunc(attrn(&dsid,NVARS)) ;
    %let rc    = %sysfunc(close(&dsid)) ;
    %put dans le fichier &ds il y a
         &nblig ligne(s) et &nbcol colonne(s) ;
    %mend asg ;
```

4.4 Somme en ligne, sieste en colonne

La macro suivante permet de rajouter une colonne à un *dataset*, nommée somme et qui contient la somme de toutes les valeurs numériques de la ligne. L'instruction array valr(*) _numeric_ ; crée un tableau nommé valr de dimension non fixée par avance, qui correspond aux valeurs numériques de la ligne. L'instruction nbcoln = dim(valr) ; affecte le nombre de ces valeurs à la variable nbcoln et une simple boucle pour suffit alors à calculer la valeur de somme.

```
%macro somlig(fid) ;
data somlig ; set &fid ;
array valr(*) _numeric_ ;
nbcol = dim(valr) ;
somme = 0 ;
do i = 1 to nbcol ;
   somme = somme + valr(i) ;
end ;
drop i nbcol ;
;
%mend somlig ;
```

Pour combiner des statistiques *de modalité* avec les données originales, par exemple pour indiquer à coté de la colonne quel pourcentage du total elle reprente, il faut procéder en général en quatre étapes :

1 - définir les données de base avec data
2 - trier les données par modalité sort...by
2 - effectuer les calculs avec means
4 - effectuer la fusion avec data option merge..by

Par contre, pour intégrer des données sans variable commune (comme un total général) il faut utiliser des techniques de set suivant le modèle

```
data  Nouveau;
      if _n_ = 1 then set Stat ;
      set Ancien ;
```

où Ancien est le *dataset* original, Stat le *dataset* des résultats (produit par proc means ... out = et Nouveau le *dataset* qu'on veut obtenir. On pourra consulter le chapitre 6 du *The little sas book* conseillé dans la bibliographie pour plus d'informations.

Essayons maintenant de calculer les totaux par colonne. Nous prendrons une approche radicalement différente de celle utilisée pour les sommes en ligne. On effectue le travail en trois étapes :

1 - on calcul les sommes avec proc means ...sum
2 - on transpose les résultats avec proc transpose
2 - on concatène les fichiers via date AetB ; set A B ;

Bien sûr il vaut mieux ne pas afficher les numéros d'enregistrements dans le nouveau *dataset*, soit les instructions

```
proc means data=<dataset> sum ; var _numeric_ ; output out=stat sum=  ;
proc transpose data = stat  out = tats ;
data ajout ; set stat ; id = 'somme ' ;
data global ; set <SATASET> ajout ; drop _type_ _freq_ ;
proc print data=global noobs;
```

4.5 Matrice des corrélations triangulaire

La procédure corr de *sas* générère une matrice des corrélations rectangulaire. L'habitude en statistiques veut que l'on ne donne que la matrice triangulaire inférieure.

Nous en profitons au passage pour montrer

comment tester si un *dataset* existe	avec sysfunc(open(
comment renommer une variable	avec rename
comment extraire des lignes	avec where
comment typer explicitement des variables	avec attrib
comment mettre les données dans un fichier externe	avec file

```
/*****************************************
 *                                       *
 * Matrice des correlations triangulaire *
 *                                       *
 *****************************************/

  %macro matcor(ds) ;

  %let donnees = &ds ;
  %let dsid    = %sysfunc(open(&donnees)) ;
  %if &dsid=0 %then %do ;
      %put LE FICHIER DE DONNEES &donnees EST INACCESSIBLE !? ;
  %end ;
  %else %do ;

  %let nbcol = %sysfunc(attrn(&dsid,NVARS)) ;

  proc corr data=&ds noprint out=scorr ;
  data mcor ; set scorr ; where _type_ = 'CORR' ;
       rename  _name_ = nom ; drop _type_ ;

  data mcortr ; set mcor ;
  array cor(*) _numeric_ ;
  attrib rho length = $200 ;
  attrib v   length = $200 ;
  do i = 1 to _n_ ;
     if i = 1 then
        do  ;  w = int(cor(1)*100+0.5)/100 ;
               v = put(w,6.3) ;
        end ;
        else  do ;  w = int(cor(i)*100+0.5)/100 ;
                    t = put(w,6.3) ;
                    v = trim(v) || ' ' || t ;
        end ;
  end ;
  rho = v ;
  keep nom rho ;
```

```
data lnom ; set scorr ; where _type_ = 'CORR' ;
     rename  _name_ = ndl ; drop _type_ ;

data lnoml  ; set lnom ;
attrib noml length = $015 ;
attrib ldv  length = $200 ;
noml = trim(ndl) || '________' ;
noml = substr(noml,1,6) ;
if _n_ = 1 then ldv = noml ;
           else ldv = trim(ldv) || ' ' || noml ;
ndl = ' '  ;
retain ldv ;
keep   ndl ldv ;

data premlig ;
     set lnoml ;
     rename ndl = nom ;
     rename ldv = rho ;
     if _n_ = &nbcol - 1 ;

data matcor ; set premlig mcortr ;

data _null_ ; set matcor ;
     file 't:matcor.sor' ;
     rhot = ' : ' || rho ;
     put nom @10 rhot $ ;

proc print data=matcor noobs ;

run ;
%put Vous pouvez consulter le fichier 'matcor.sor' ;

%end ; /* fin de si le fichier de donnees existe */
run ;

%mend matcor ;
```

Le *dataset* particulier _null_ évite à *sas* de sauvegarder le *dataset* utilisé, ce qui peut économiser de la mémoire et de l'espace disque...

Chapitre 5.

Tracés et Graphiques en SAS

SAS dispose de nombreux modes et procédures de tracé. Le débutant est souvent perdu (déçu ?) par la complexité des instructions à mettre en oeuvre pour un simple tracé, surtout par comparaison avec Excel, Statistica ou Gnuplot. Par contre, dés qu'on a une série de graphiques à effectuer sur le même modèle, SAS se révéle d'un usage extraordinaire.

5.1 Modes et procédures de tracé

SAS connait les modes de tracé TEXTE et HAUTE RESOLUTION. En mode texte, les grahiques sont placés dans le fichier d'OUTPUT, comme du texte normal. Les graphiques y sont réalisés à l'aide de caractères normaux et ressemblent à ce que produirait une machine à écrire. L'avantage de ce mode est d'être totalement portable car il ne s'agit que du texte. Les graphiques sont grossiers (par rapport à un affichage écran) mais suffisant dans la plupart des cas. En mode haute résolution, les graphiques sont affichés dans des fenêtres spéciales nommés GRAPHi WORK.GSEG.XXXj où XXX est la procédure de tracé utilisée et j le numéro du graphique en cours. A l'aide du menu Général/Graphiques on peut revoir tous les tracés effectués, les modifier avec un éditeur graphique élémentaire, les exporter EN BMP, GIF, JPG, PS etc.

Pour tester l'affichage en haute résolution, c'est à dire surtout l'adéquation du fichier de configuration de SAS au moniteur, on utilise l'instruction :

```
proc gtestit ; run ;
```

De nombreuses procédures de tracé en mode texte comme PLOT, CHART ont leur équivalent en haute résolution : il suffit d'ajouter un g en début de procédure, comme par exemple GPLOT et GCHART. La procédure GDEVICE permet de définir le pilote (DRIVER) d'écran haute résolution.

5.2 Généralités sur les tracés en SAS

Les procédures de tracé ont souvent de nombreuses options propres, mais aussi des options locales qui portent le même nom que des options globales. Ainsi TITLE en tant qu'instruction avant PLOT est une option globale qui mettra le titre indiqué pour tous les graphiques alors que TITLE en tant qu'option de la procédure PLOT ne changera le titre que pour le tracé correspondant. En termes d'instuctions :

```
     TITLE  'global ' ;
     proc PLOT ;
          ....  /* le titre est global */
     proc PLOT ;
          ....  /* le titre est global */
/************************************/
     proc PLOT ;
          TITLE  'local 1' ;
          ....  /* le titre est local 1*/
     proc PLOT ;
          TITLE  'local 2' ;
          ....  /* le titre est local 2 */
```

De nombreuses options sont communes comme SYMBOLn, TITLEn, PATTERNn où le n indique qu'on mettre plusieurs fois l'instruction avec un numéro de tracé, voire un facteur de répétition avec REPEAT (ou R en raccourci). Ainsi

```
SYMBOL1 c=red ; SYMBOL2 c=blue ;
```

indique que la première courbe sera tracé en rouge, la seconde en bleu et

```
SYMBOL1 c=red r=4; SYMBOL2 c=blue ;
```

indique que les 4 premières courbes seront tracées en rouge, la cinquième en bleu.

5.3 La procédure PLOT

Lorsqu'on trace des courbes, le modèle d'écriture Y*X=Z signifie "tracer la courbe Y en fonction de X avec le caractère Z". Pour tracer, à l'intérieur de l'instruction PLOT, on utilise l'option PLOT (si, si, deux fois PLOT). Par exemple :

```
data gr1 ;
input x y z f i $ 20-22;
datalines ;
 100  125  200  1  Jan
 200  200  390  1  Fev
 300  375  410  1  Mar
 500  450  500  2  Sep
 600  500  550  2  Oct
 700  125  200  1  Nov
;

proc plot data=gr1 ;
     plot  y*x ;

run ;
```

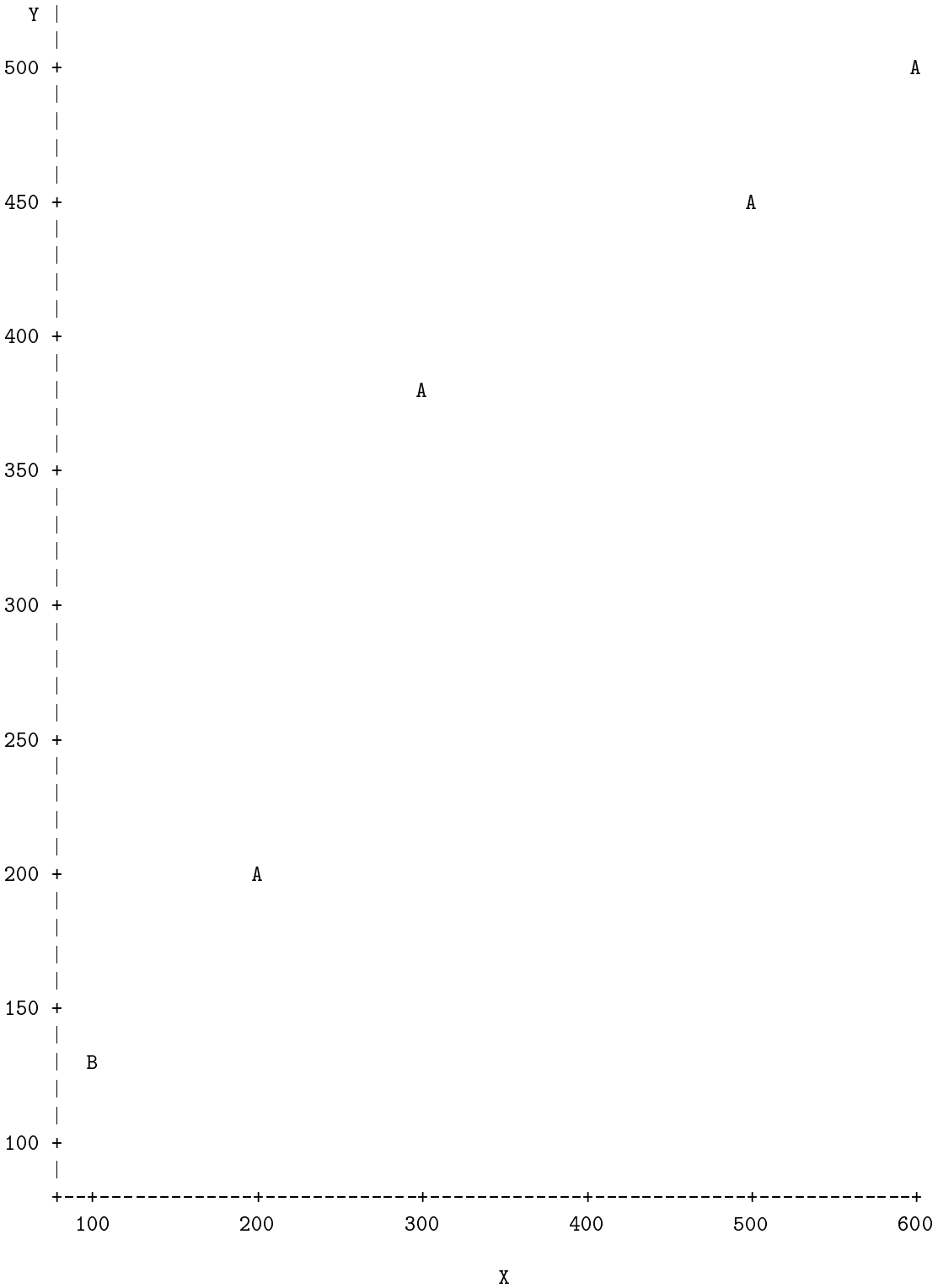
Puisqu'ici on n'a pas indiqué de symbole de tracé, SAS affiche en mode texte avec le caractère A pour indiquer le point à tracer s'il est unique, B s'il y a deux valeurs, etc. Pour notre exemple, A est donc tracé à la coordonnée (100,125). Si au lieu de la dernière ligne

```
700  125  200  1  Nov
```

nous avions mis

```
100  125  200  1  Nov
```

nous aurions eu un B à la coordonnée (100,125) comme sur le graphique de la page suivante.


Y
500
450
400
350
300
250
200
150
100
A
A
A
A
B
100
200
300
400
500
600
X

Lorsque RUN est entré à la suite de l'instruction PLOT, le tracé est réalisé et la procédure PLOT est en cours d'utilisation. Il faut alors entrer QUIT ; pour terminer la procédure QUIT avant d'utiliser une autre procédure.

Pour tracer plusieurs courbes, on utilise plusieurs fois le modèle Y*X. La sous-option OVERLAY (les sous-options commencent par /) produit les tracés sur le même graphique. Par exemple `plot y*x z*x` trace y en fonction de x sur un graphique puis z en fonction de x sur un autre graphique alors que `plot y*x z*x / overlay` trace y et z en fonction de x sur le même graphique.

Grâce aux options hpercent et vpercent, on peut mettre plusieurs graphiques sur une même page. Ainsi

```
proc plot data=gr1 hpercent=50 ;
     plot y*x z*x ;
```

affiche deux courbes distinctes sur une même page, l'une à coté de l'autre alors qu'avec vpercent on les aurait eu l'une au-dessus de l'autre. La combinaison `hpercent=50 vpercent=50` permet donc de produire 4 graphiques sur une même page en proportions égales et la combinaison `hpercent=25 vpercent=20` fournit 5 lignes de 4 graphiques sur une seule page (il est déconseillé de descendre en dessous de `hpercent=25` %).

La sous-option `box` trace un cadre autour-du graphique. Pour rajouter une ligne de référence, on entre les sous-options `href` ou `vref`. Par exemple `vref=400` produit une ligne horizontale à la hauteur y=400. Il reste d'autres options à exploiter puisque la syntaxe complète de PLOT est

```
PROC PLOT <option-list>;
     PLOT request-list </ suboptions >;
     BY   variable-list;
```

où les options sont

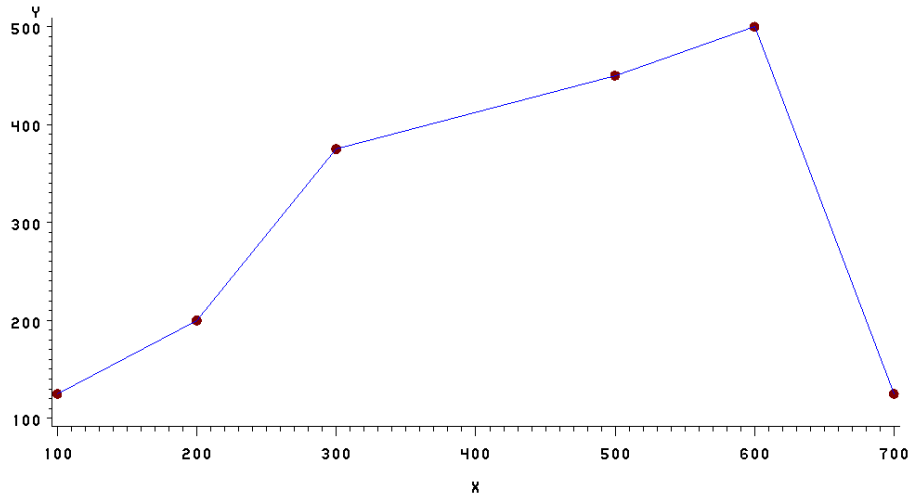
```
Input Data Set   DATA= SAS-data-set
Number of Plots  HPERCENT= percent-list     VPERCENT= percent-list
Missing Values   MISSING                    NOMISS
Appearance       NOLEGEND                   UNIFORM
Aspect Ratio     VTOH= character-height/character-width
Plot Border      FORMCHAR <(index-list)>= 'formchar-string'
```

mais nous renvoyons au manuel et à la documenation en ligne pour le détail de l'ensemble de l'instruction PLOT.

5.4 La procédure GPLOT

Passons maintenant à la procédure GPLOT, "grande soeur" de PLOT. Commençons par tracer deux courbes sur le même graphique, la première en rouge avec des points (x,y) représentés par des cercles et la seconde en bleu avec des segments pour joindre les points (x,y) sans symbole de point :

```
symbol1 c=red  v=dot ;
symbol2 c=blue i=join ;
proc gplot data=gr1 ; plot y*x y*x / overlay ;
```



La syntaxe complète de GPLOT est

```
PROC GPLOT options;
     PLOT      yvariable*xvariable... / options;
     PLOT2     yvariable*xvariable... / options;
     BUBBLE    yvariable*xvariable=zvariable;
     BUBBLE2   yvariable*xvariable=zvariable;
     BY        variables;
     AXIS      options;
     LEGENDn   options;
     SYMBOLn   options;
     PATTERNn  options;
     TITLEn    options 'text';
     FOOTNOTEn options 'text';
     NOTE      options 'text';
```

Et il y a de nombreuses sous-options. Par exemple pour PLOT et PLOT2 :

```
OVERLAY                 CHREF=  color          HMINOR= n
AREAS= n                LHREF=  linetype       NOAXIS
SKIPMISS                VREF=  values          VZERO
LEGEND=  LEGENDn        CVREF=  color          HZERO
NOLEGEND                LVREF=  linetype       FRAME
ANNOTATE=  SASdataset   VAXIS=  values|AXISn   CFRAME= color
DESCRIPTION= 'string'   HAXIS=  values|AXISn   CAXIS= color
NAME= 'string'          VREVERSE               CTEXT= color
HREF=  values           VMINOR= n              GRID
AUTOHREF                AUTOVREF               HREVERSE
```

alors que les options pour BUBBLE et BUBBLE2 sont

```
ANNOTATE= Annotate-data-set     GRID
AUTOHREF                        HAXIS= value-list | AXIS<1...99>
AUTOVREF                        HMINOR= n
BCOLOR= bubble-color            HREF= value-list
BFONT=  font                    HZERO
BLABEL                          LHREF= line-type
BSCALE= AREA|RADIUS             LVREF= line-type
BSIZE=  multiplier              NAME=  'string'
CAXIS= axis-color               NOAXIS
CFRAME= background-color        VAXIS= value-list | AXIS<1...99>
CHREF=  reference-line-color    VMINOR= n
CTEXT= text-color               VREF= value-list
CVREF= reference-color          VREVERSE
DESCRIPTION=  'string'          VZERO
FRAME
```

Mais il ne faut pas se laisser abuser par cette profusion d'options et de sous-options : elles sont toutes utiles et assez faciles à utiliser. Ainsi l'utilisation des symboles numérotés avec SYMBOLn permet de donner un style de tracé : c donne la couleur des lignes, v la valeur du symbole à mettre pour chaque point, cv est la couleur de ce symbole, i est l'interpolation entre deux points : vide, les points sont isolés ; avec i=join les points sont reliés par un segment de droite. L'épaisseur des traits est gérée par w (comme width), le style de ligne (plein ou pointillé) par L. Les autres options (comme spline, needle...) sont affichées en fin de chapitre.

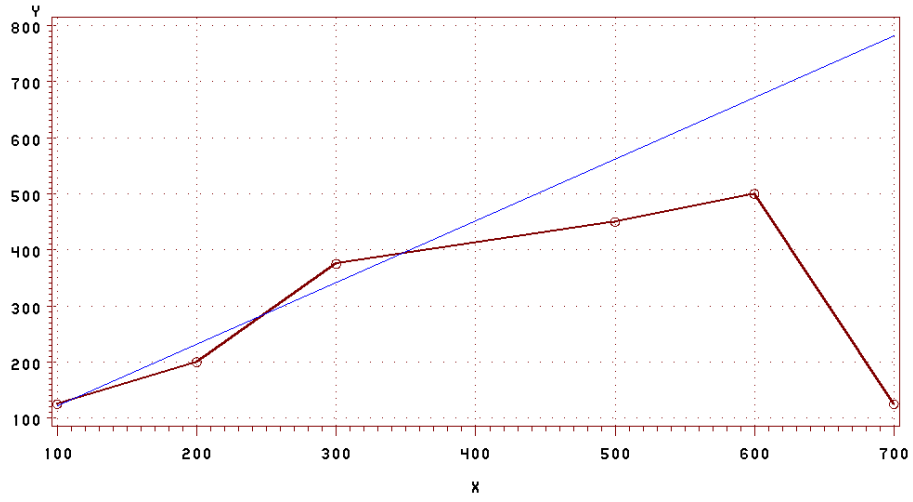
Imaginons que nous ayons besoin de tracer une droite de référence à coté de nos points, par exemple la droite d'équation $y = 1.1x + 12$. A l'aide d'une procédure DATA, nous rajoutons des points à notre ensemble de données puis nous traçons cette droite en plus dans notre graphique. Nous en profitons

pour utiliser la sous-option GRID et nous changeons au passage la couleur des axes avec CAXIS :

```
data gr2 ;
set gr1 ;
yc = x*1.1 + 12 ;

symbol1 c=red  v=circle i=join width=2;
symbol2 c=blue v=none   i=join width=1;
symbol3 c=green ;

proc gplot data=gr2 ;
     plot y*x yc*x /
     overlay caxis=red grid chref=red ;
```



On trouvera dans la section *Tracés vraiment statistiques* une utilisation intensive et plus poussée des différents paramètres.

5.5 La procédure GCHART

Les commandes CHART et GCHART tracent des histogrammes sous forme de barres horizontales avec l'option HBAR ou verticales avec l'option VBAR.

```
PROC GCHART   options;
     HBAR      variables... / options;
     VBAR      variables... / options;
```

```
BLOCK     variables    / options;
PIE       variables... / options;
STAR      variables... / options;
DONUT     variables... / options;
BY        variables;
AXISn     options;
LEGENDn   options;
PATTERNn  options;
TITLEn    options 'text';
FOOTNOTEn options 'text';
NOTE      options 'text';
```

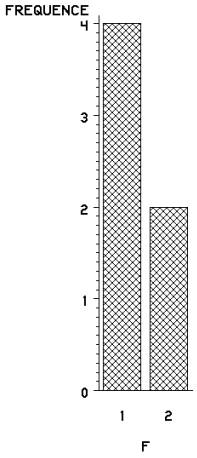
Essayons par exemple de tracer des histogrammes de comptages et de fréquences pour la colonne F du *dataset* GR1 :

X	Y	Z	F	I
100	11.250	200	1	Jan
200	39.200	390	1	Fev
300	14.750	410	1	Mar
500	45.500	500	2	Sep
600	62.000	550	2	Oct
700	56.250	700	1	Nov

SAS sait tracer des histogrammes pour des variables qualitatives et quantitatives. Pour des variables quantitatives, SAS trace des classes avec indication de la valeur de centre de classe. Comme nos données correspondent à une variable qualitative, nous utilisons la sous-option DISCRETE et SAS fournit directement les fréquences absolues (nommées simplement FREQUENCES par SAS) ou relatives (nommées POURCENTAGES).

Le graphique de base utilise un quadrillage pour les barres avec un maximum d'options par défaut :

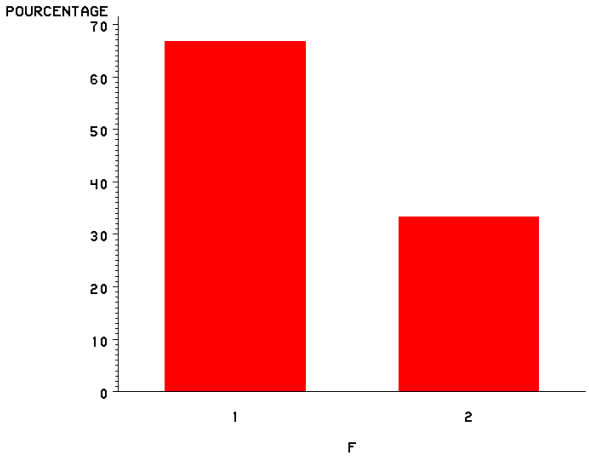
```
proc gchart data=gr1 ;
     vbar f / discrete  ;
```



Pour afficher l'histogramme des fréquences relatives (POURCENTAGES), on utilise la sous-option TYPE. De plus, nous décidons de remplir (s pour "SOLIDE") en rouge orange chaque barre avec l'option PATTERN, de laisser 10 unités entre les barres qui auront comme largeur 15 unités :

```
proc gchart data=gr1 ;
     vbar f / discrete  type=percent width=15 space=10 ;
     pattern2 v=s c=orange ;
```

soit le graphique :



La richesse des sous-options de VBAR mérite attention et de nombreuses heures d'entrainement pour les maitriser :

ALPHACLM	ANNOTATE	ASCENDING	AUTOREF
AXIS	CAXIS	CERROR	CFRAME
COUTLINE	CTEXT	DESCENDING	DESCRIPTION
DISCRETE	ERRORBAR	FRAME	FREQ
G100	GAXIS	GROUP	GSPACE
LEGEND	LEVELS	MAXIS	MIDPOINTS
MINOR	MISSING	NAME	NOAXIS
CFREQ	NOLEGEND	NOZERO	PATTERNID
RAXIS	REF	SPACE	SUBGROUP
SUMVAR	TYPE	WIDTH	CLIPREF
CPERCENT	MEAN	PERCENT	SUM
NOBASEREF			

Nous allons essayer d'en utiliser quelques-unes avec l'exemple du dossier ELF. Dans ce dossier, des hommes et des femmes ont rempli un questionnaire et en particulier ils ont indiqué leur niveau d'études avec un code :

```
0 = pas de niveau fourni
1 = niveau Primaire
2 = niveau Secondaire
3 = niveau Bac
4 = niveau Sup.
```

On commence bien sûr par utiliser la procédure format pour donner des libellés lisibles aux codes :

```
proc format ;
     value f_sexe    0='Homme' 1='Femme' ;
     value f_etudes  0='NR'
                     1='Primaire'
                     2='Secondaire'
                     3='Bac'
                     4='Sup' ;
```

Le tri-croisé que produit SAS avec les instructions

```
data elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18 ETUDES 23 ;
 format SEXE f_sexe. ETUDES f_etudes. ;
datalines ;
M001    1  62    1    2
M002    0  60    9    3
...
M100    1  48    9    4
;
proc freq data=elf ;
     table etudes*sexe / norow nocol nofreq ;
```

est intéressant mais peu visuel :

	Hommes	Femmes	Total
NR	2.02	1.01	3.03
Primaire	1.01	5.05	6.06
Secondaire	7.07	23.23	30.30
Bac	8.08	13.13	21.21
Sup	17.17	22.22	39.39
Total	35.35	64.65	100.00

C'est pourquoi nous allons adjoindre des graphiques mais attention : seuls des histogrammes bien choisis facilitent la lecture du tri croisé (un histogramme mal choisi est ici soit trivial soit inutile).

Notre but est de montrer ce qu'une lecture attentive du tri croisé révèle : s'il y a plus de femmes que d'hommes et beaucoup de personnes ayant fait des études supérieures, les hommes et les femmes n'ont pas la même profil d'études.

Nous avons le choix entre deux techniques pour tracer les histogrammes : tracer chacun avec son échelle, de façon à utiliser un espace maximal pour les barres, ou utiliser des axes communs et une même échelle.

Souvent les logiciels classiques (*Excel* par exemple) ne permettent pas de définir une échelle commune. Avec SAS c'est un jeu d'enfant : pour tracer deux histogrammes avec les mêmes options, il suffit d'indiquer les mêmes paramètres. En l'occurence, les histogrammes de fréquences séparés de la variable SEXE et de la variable ETUDES sont tracés avec les mêmes options pour en faciliter la comparaison grace à l'option RAXIS et nous en profitons au passage pour mettre une barre de référence (REF=) à la hauteur 50 % pour indiquer une modalité majoritaire, que nous demandons à SAS de masquer derrière la barre (CLIPREF). Ainsi

```
proc gchart data=elf ; pattern v=s c=blue ;
     vbar etudes / discrete ref=50 frame raxis=axis1 clipref ;
     axis1 order=(0 to 100 by 20) minor=none label=none ;
```

et

```
proc gchart data=elf ; pattern v=s c=blue ;
     vbar etudes / discrete ref=50 frame raxis=axis1 clipref ;
     axis1 order=(0 to 100 by 20) minor=none label=none ;
```

produisent des histogrammes à même échelle puisque la spécification des axes est la même. En fait, SAS propose carrément de mettre la LISTE des variables à tracer dans ce cas-là et donc la "bonne" instruction est

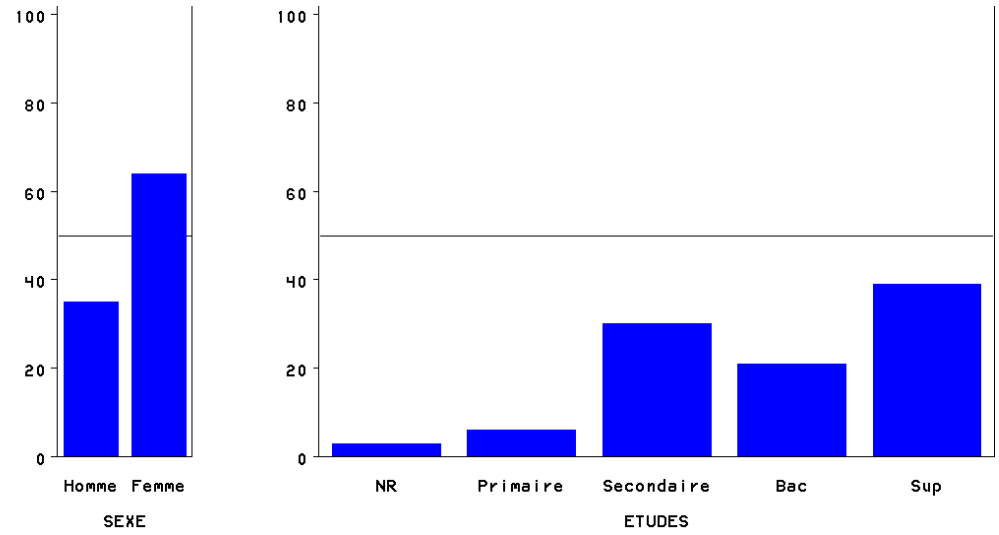
```
proc gchart data=elf ;
     vbar sexe etudes / discrete ref=50 frame raxis=axis1 clipref ;
     pattern v=s c=blue ;
     axis1 order=(0 to 100 by 20) minor=none label=none ;
```

Si on ne veut pas les mêmes axes, il suffit de ne rien dire et de laisser SAS choisir seul :

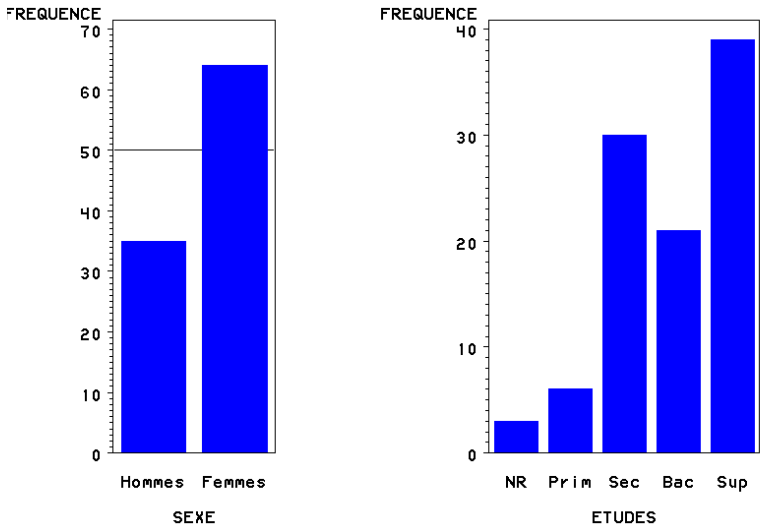
```
proc gchart data=elf ; pattern v=s c=blue ;
     vbar sexe etudes / discrete ref=50 frame clipref ;
```

On trouvera sur la page suivante les tracés correspondants. Il faut signaler que SAS ne permet pas de tracer plusieurs histogrammes sur une même page. C'est donc par la mise en page (sous *Word*, *LaTeX*) qu'on produit le résultat précédent.

```
proc gchart data=elf ;
     vbar sexe etudes / discrete ref=50 frame raxis=axis1 clipref ;
     pattern v=s c=blue ;
     axis1 order=(0 to 100 by 20) minor=none label=none ;
```



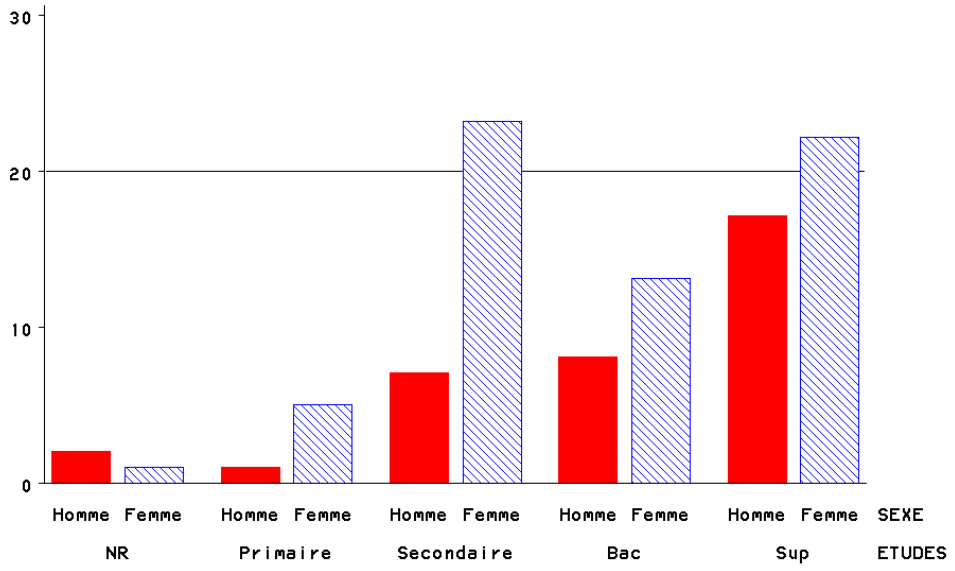

```
proc gchart data=elf ;
     vbar sexe etudes / discrete ref=50 frame clipref ;
     pattern v=s c=blue ;
```



Clairement, sans axes communs, la lecture des graphiques est un peu moins immédiate car les échelles ne sont pas les mêmes sur les deux graphiques. Passons maintenant à l'histogramme de notre tri croisé. Nous avons là encore le choix entre réaliser l'histogramme des niveaux d'études pour chaque code-sexe et l'histogramme des sexes pour niveau d'étude. SAS dipose de l'option BY des sous-options GROUP ET SUBGROUP dans l'option VBAR.

L'histogramme des niveaux d'études par sexe ne fait que répéter la présence de deux fois plus de femmes que d'hommes :

```
proc gchart data=elf ;
   vbar sexe / type=percent discrete raxis=axis1
               group=etudes subgroup=sexe legend=legend1 ;
   pattern1 v=s c=orange ;
   pattern2 v=L1 c=blue ;
   legend1 value=none label =none ;
   axis1 order=(0 to 30 by 10) minor=none label=none ;
```

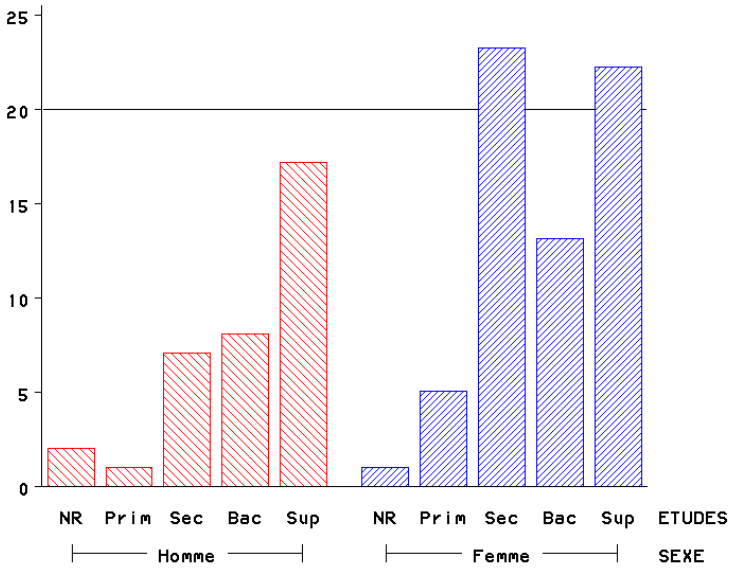


Maintenant, si nous voulons batir l'histogrammes de la variable SEXE par niveau d'étude, il faut changer de format car sinon les mots Primaire, les mots Secondaire seront trop longs et SAS les affichera verticalement plutot qu'horizontalement.

On voit alors distinctement que les hommes et les femmes n'ont pas le même profil d'études :

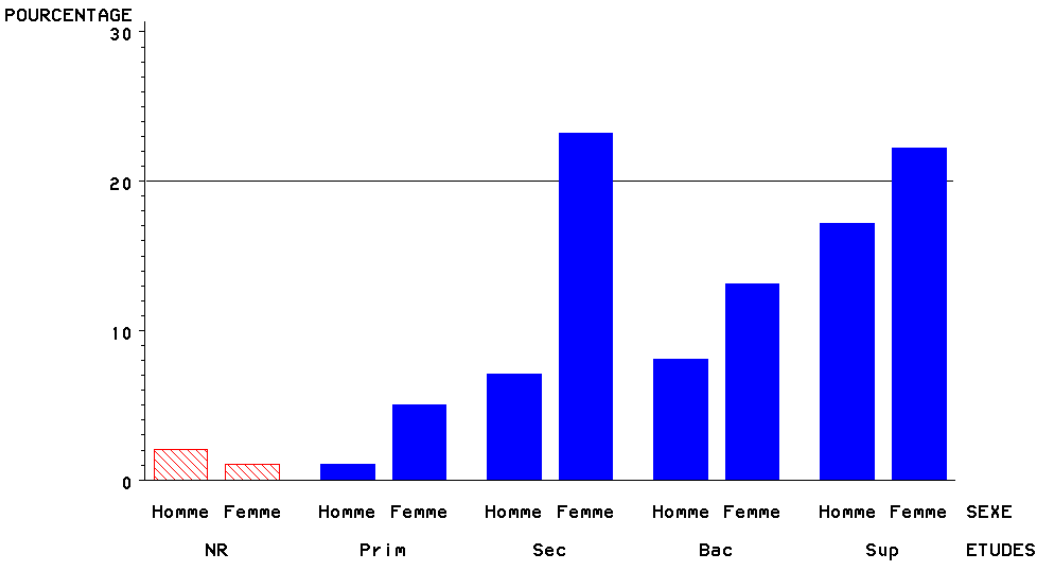
```
proc format ;
     value f_sexe    0='Homme' 1='Femme' ;
     value f_etudes  0='NR'
                     1='Prim'
                     2='Sec'
                     3='Bac'
                     4='Sup' ;
data elf2 ;
 set elf ;
 input ID $ 1-4 SEXE 9 AGE 12-13 PROF 17-18 ETUDES 23 ;
 format SEXE f_sexe. ETUDES f_etudes. ;
;

  proc gchart data=elf ;
     vbar etudes / type=percent group=sexe subgroup=sexe discrete
                   ref=20 raxis=axis1 legend=legend1 nolegend clipref ;
     pattern1 v=L1 c=orange ;
     pattern2 v=R2 c=blue ;
     axis1 order=(0 to 25 by 5) minor=none label=none ;
     legend1 value=none label =none ;
run ; quit ;
```



On peut aller un cran plus loin dans l'affichage avec le facteur de répétition r= de l'option PATTERN : les données manquantes par non réponse (code 0) vont être affichées différemment des autres valeurs :

```
proc gchart data=elf ;
   vbar sexe / type=percent group=etudes subgroup=etudes discrete ;
   pattern1 v=L1 c=orange r=1;
   pattern2 v=s  c=blue r=43;
```



5.6 Tracés vraiment statistiques

Lorsqu'on effectue des calculs de régression linéaire, il est d'usage d'afficher les données, la droite de régression, les valeurs hautes et basses liées à l'intervalle de confiance à 95 %, la distribution des résidus etc. Pour un logiciel classique, comme Excel, cela oblide à de nombreux calculs, voire à programmer des formules. SAS sachant faire les calculs correspondants, il suffit en génral de lui demander de stocker les résultats plutot que des les afficher pour ensuite réutiliser les variables pour des tracés. Montrons cela sur un exemple : à la suite de l'instruction DATA, nous effectuons une régression linéaire élémentaire avec l'instruction REG sans rien afficher (option NOPRINT) mais en stockant les résultats (option OUTPUT, sous-option OUT) dans le *dataset* nommé Gr_REG en renommant au passages les bornes des valeurs de confiance à 5 % en HAUT et BAS.

Voici donc tout d'abord les instructions pour effectuer la régression et stocker les résultats :

```
data gr5 ;
input x y z f i $ 20-22;
datalines ;
 100  11.250  200  1  Jan
 200  39.200  390  1  Fev
 300  14.750  410  1  Mar
 500  45.500  500  2  Sep
 600  62.000  550  2  Oct
 700  56.250  700  1  Nov
;


proc reg data=gr5 noprint ;
     model y=x ;
     output out=gr_reg
            predicted=yp
            redisual=res
            u95m=haut
            l95m=bas ;
```

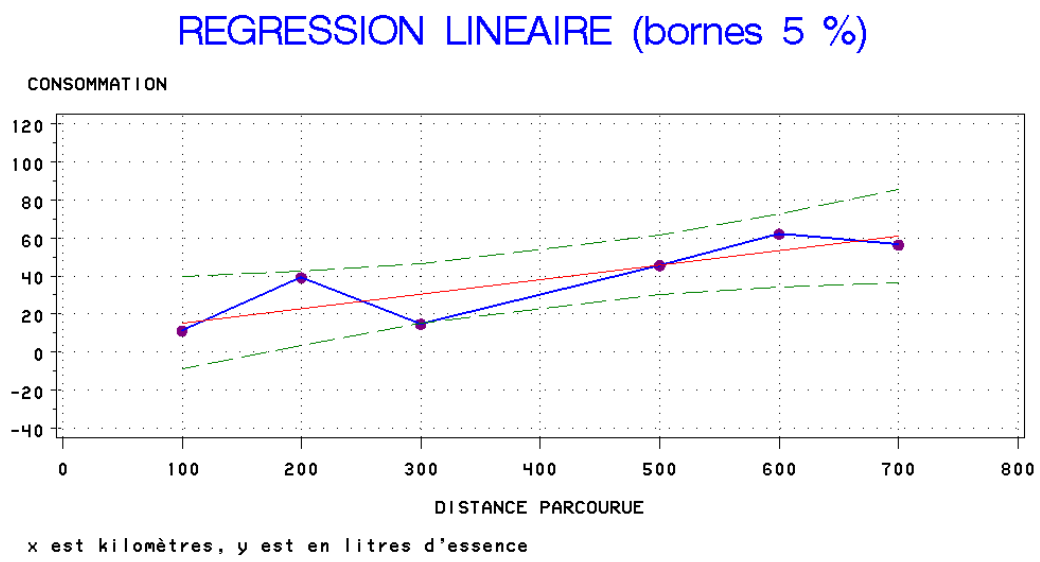
Traçons maintenant avec GPLOT ce qui nous intéresse et pour une fois nous gérons le titre avec la police de caractères sas monospace bold et nous rajoutons des notes au graphique :

```
proc gplot data=gr_reg ;

     symbol1 c=blue v=dot cv=magenta i=join w=2;
     symbol2 c=orange i=join ;
     symbol3 c=green l=2 r=2 i=join;
     title1 c=blue 'REGRESSION LINEAIRE (avec bornes 95 %)' ;
     title2 ' ' ;
     title3 c=black j=left '  CONSOMMATION' ;
     footnote1 j=left '  x est km, y est en litres d''essence' ;
     footnote2 j=left ' ' ;
     axis1 order=(0 to 800 by 100) minor=none label=('DISTANCE PARCOURUE') ;
     axis2 order=(-40 to 120 by 20) minor=(number=1) label=none  ;

     plot y*x
          yp*x
          haut*x
          bas*x
     / overlay grid haxis=axis1 vaxis=axis2 frame ;
```

Le résultat est alors



Comme deuxième exemple de graphique statistique, nous montrons la superposition de la loi beta à un histogramme. Cet exemple provient de la SAS Sample Libary qui est un ensemble de programmes SAS écrits à titre d'exemples et accessibles par le menu d'aide sous la rubrique Exemples de programmes dans la configuration française.

Nous avons choisi cet exemple parce qu'il montre que des procédures autres que GPLOT et GCHART savent tracer des graphiques et parce qu'il utilise des options graphiques générales (GOPTIONS plutot que OPTIONS), notamment celle qui permet de réinitialiser mes options graphiques (en fin de programme).

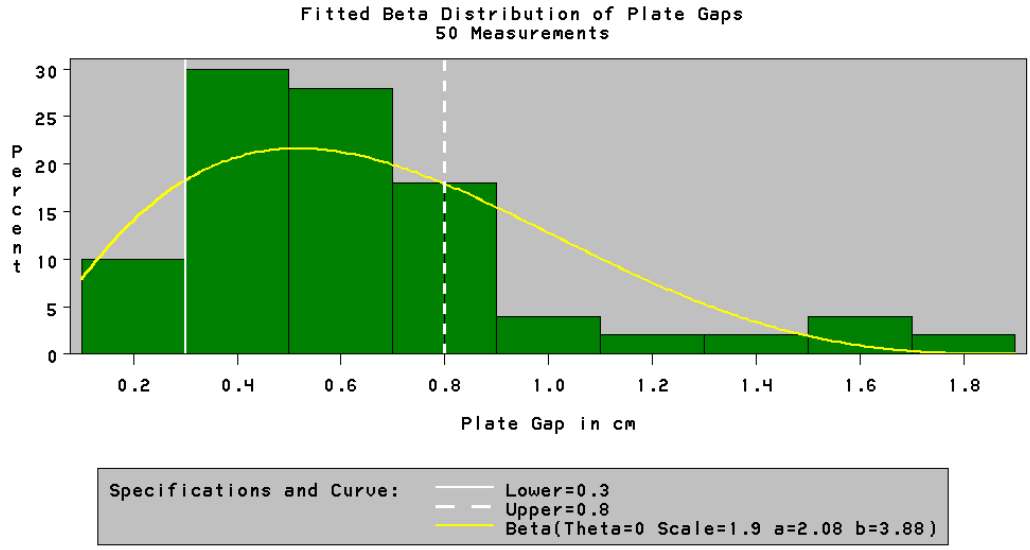
```
/*************************************************************/
/*                                                           */
/*    S A S   S A M P L E   L I B R A R Y                    */
/*                                                           */
/*    NAME    : CAPBTA2                                      */
/*    TITLE   : Fitting a Beta Curve on a Histogram          */
/*    PRODUCT : QC                                           */
/*                                                           */
/*************************************************************/

goptions ftext=none htext=1 cell;

data plates;
   label gap='Plate Gap in cm';
   input gap @@;
   cards;
0.746  0.357  0.376  0.327  0.485
1.741  0.241  0.777  0.768  0.409
0.252  0.512  0.534  1.656  0.742
0.378  0.714  1.121  0.597  0.231
0.541  0.805  0.682  0.418  0.506
0.501  0.247  0.922  0.880  0.344
0.519  1.302  0.275  0.601  0.388
0.450  0.845  0.319  0.486  0.529
1.547  0.690  0.676  0.314  0.736
0.643  0.483  0.352  0.636  1.080
;
```

```
title1 'Fitted Beta Distribution of Plate Gaps';
title2 '50 Measurements';

proc capability data=plates noprint graphics;
   var gap;
   specs lsl=0.3 clsl=white llsl=1  wlsl=2
         usl=0.8 cusl=white lusl=20 wusl=2;
   histogram /
      beta(theta=0 scale=1.9 color=yellow w=2)
      midpoints = 0.2 to 1.8 by 0.2
      legend    = legend1
      cframe    = gray
      cfill     = green;
   legend1 cframe=gray cborder=black;
run;
goptions reset=all;
```



5.7 Détails des options des procédures

```
AXIS <1...99>
   COLOR|C   = axis-color
   LABEL     = (<text-description|'text'>
                <...text-description-n|...'text-n'>)|NONE
   LENGTH    = n<units>
   LOGBASE   = base|E|PI
   LOGSTYLE  = EXPAND|POWER
   MAJOR     = (tick-mark-description)|NONE
   MINOR     = (tick-mark-description)|NONE
   NOBRACKETS
   OFFSET    = (<n1><,n2>)<units>|(<n1<units>><,y<units>>)
   ORDER     = (value-list)
   ORIGIN    = (<x><,y>)<units>|(<x<units>><,y<units>>)
   STYLE     = line-type
   VALUE     = (<text-description|'text'>
                <...text-description-n|...'text-n'>)|NONE
   WIDTH     = n;

LEGEND <1...99>
   ACROSS    = n
   CBLOCK    = block-color
   CBORDER   = frame-color
   CFRAME    = background-color
   CSHADOW   = shadow-color
   DOWN      = n
   FRAME
   LABEL     = (<text-description|'text'>
                <...text-description-n|...'text-n'>)|NONE
   MODE      = RESERVE|SHARE|PROTECT
   OFFSET    = (<x><,y>)<units>|(<x<units>><,y<units>>)
   ORIGIN    = (<x><,y>)<units>|(<x<units>><,y<units>>)
   POSITION  = (<BOTTOM|MIDDLE|TOP> <LEFT|CENTER|RIGHT>
                <OUTSIDE|INSIDE>)
   SHAPE     = BAR(width,height)<units>|LINE(length)<units>|
               SYMBOL(width,height)<units>
   VALUE     = (<text-description|'text'>
                <...text-description-n|...'text-n'>)|NONE;
```

```
   PATTERN <1...99>
      COLOR|C   = pattern-color
      REPEAT|R  = n
      VALUE|V   = bar/block-pattern | map/plot-pattern | pie/star-pattern;

   SYMBOL <1...99>
      CI         = line-color
      CO         = color
      COLOR|C    = symbol-color
      CV         = value-color
      FONT|F     = font
      HEIGHT|H   = n<units>
      INTERPOL|I = BOX<options><0...25>
      INTERPOL|I = HILO<C><options>
      INTERPOL|I = JOIN
      INTERPOL|I = L<degree><P><S>
      INTERPOL|I = map/plot-pattern
      INTERPOL|I = NEEDLE
      INTERPOL|I = NONE
      INTERPOL|I = R<type><0><CLM|CLI<50...99>>
      INTERPOL|I = SM<nn><P><S>
      INTERPOL|I = SPLINE<P><S>
      INTERPOL|I = STD<1|2|3><variance><options>
      INTERPOL|I = STEP<placement><J><S>
      LINE|L     = line-type
      MODE       = EXCLUDE|INCLUDE
      REPEAT|R   = n
      VALUE|V    = special-symbol|text-string|NONE
      WIDTH      = n;

TITLEn options 'text';
The following options can be specified in the TITLE  statement:

      ANGLE           #BYVAL        JUSTIFY
      BCOLOR          #BYVAR        LANGLE
      BLANK           COLOR         LSPACE
      BOX             DRAW          MOVE
      BSPACE          FONT          ROTATE
      #BYLINE         HEIGHT        UNDERLIN
```

Chapitre 6.

La galaxie SAS

Comme ont du le montrer certaines listes d'options, *SAS* semble tout savoir en statistiques. C'est normal, il est "créé pour". On trouve donc dans les manuels de *SAS*, et c'est indispensable, les pages de description des options, des paramètres, des exemples d'utilisation mais aussi les formules de calcul et les références bibliographiques des auteurs de ces formules ou des articles significatifs quant à leur compréhension.

SAS comprend aussi des systèmes de cartographies, de la gestion de fichiers à travers les réseaux, un système complet de développement d'applications, des gestionnaires de fichiers etc. etc. Par contre, ce qu'on peut lui reprocher c'est d'être parfois trop complet et d'obliger à lire 10 pages d'option avant de comprendre comment "filtrer" les affichages, d'être "lourd" quant à la syntaxe et la façon de procéder, de ne pas chercher la simplicité mais l'efficacité, et aussi son prix, car *SAS* ne s'achète pas, il se loue...

BIBLIOGRAPHIE

SAS dispose de sa propre maison d'édition, de ses propres ouvrages, sans compter les nombreux manuels "standards". Notre bibliographie est donc forcément incomplète.

R. P. Cody, J. K. Smith
Applied statistics and the sas programming language
Prentice-Hall, 1997.

L. D. Delwiche, S. J. Slaughter
The little sas book, a primer
Sas Institute Inc., 1995.

Sas Language
Sas Institute Inc., 1997.

Sas Stat User's Guide vol. 1 et 2
Sas Institute Inc., 1997.

Sas Procedures Guide
Sas Institute Inc., 1997.

Sas Macro Language Reference
Sas Institute Inc., 1997.