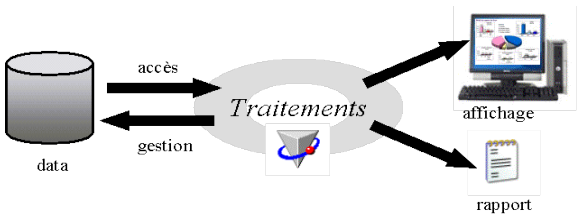
ours Logiciel SAS

Table of Contents

1 INTRODUCTION AU LOGICIEL SAS.

1 Présentation

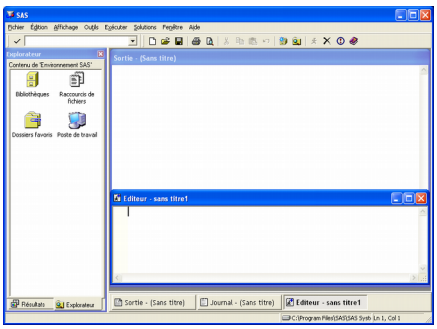
SAS est un logiciel permettant l'accès, la gestion, l'analyse et la présentation des données. Ces données peuvent aussi bien être des nombres que des caractères alpha numériques.



Plusieurs utilisations du logiciel sont disponibles : unix, windows, mode interactif, mode non interactif.

2 Environnement de travail SAS sous window

Le programme « sas.exe » permet d'accèder à l'environnement de travail SAS.



L'environnement de travail SAS sous windows présente :
Un menu (fichier, Edition…) et des fenêtres de travail :
Editeur : pour taper, charger du code SAS à exécuter…
Sortie : pour visualiser les résultats après exécution,
Journal : pour récupérer la trace d'exécution. La fenêtre Journal permet en particulier de s'assurer que les instructions SAS ne contiennent pas d'erreur syntaxique (messages Warning, error… de différentes couleurs).

3 Programme SAS

Un programme SAS est constitué d'une suite d'étapes (steps). Il existe deux étapes majeures :
- DATA : pour l'accès aux données ;
- PROC : pour les procédures de traitements des données.

Chaque étape est constituées d'une suite d'instructions se terminant par « ; ». Les commentaires sont encadrés par les caractères: /* *commentaires* */

4 Organisation des données sous SAS.

Par convention, les individus (observations) sont indiqués en lignes, les variables en colonnes. Les variables sont de deux types, numérique et alphanumérique. Les valeurs des variables de type numérique sont des nombres, et on pourra calculer leur somme, leur moyenne ...
Nous allons illustrer la manipulation des données SAS, par l'étude des données européennes.

5 Accès interactif aux données

Les instructions suivantes permettent de rentrer interactivement les données associées aux individus statistiques analysés à travers 5 variables.
Le caractère « $ » inscrit après une variable indique une donnée alphanumérique, par opposition aux données numériques.

```
/*Analyse des données européennes*/
DATA europe ;
INPUT pays $ population superficie pib date monnaie $ ;
CARDS ;
Allemagne 82 356854 23950 1957 euro
Autriche 8.1 8358 26680 1995 euro
;
RUN ;
PROC PRINT data=europe ;
RUN ;
```

- Rentrer (Editeur) et exécuter (Menu/Exécuter) les instructions SAS ci-dessus, en rentrant les données correspondant à 5 pays européens. Analyser la fenêtre Journal « .log » et la fenêtre Sortie de résultats. Corriger vos instructions si nécessaire pour obtenir une exécution correcte.
- Sauvegarder sur votre clef USB ou sur le disque dur le contenu de l'éditeur.

6 Des options pour l'affichage

Ce point concernait les versions de SAS antérieure à la 9.3. Lisez ce point puis passez à la section

suivante.
Par défaut les sorties étaient sur des pages séparées. Pour améliorer l'expérience de SAS, nous utilisions l'option suivante :

```
option formdlim='*';
```

Le mot formdlim peut se comprendre comme "forme de la délimitation".
Une fois cette option validée, elle reste valide durant toute la session SAS.

7 Accès aux données stockées dans un fichier plat:

Les instructions suivantes permettent de charger dans l'espace de travail SAS des données sauvegardées dans un fichier plat.

```
/*Analyse des données européennes, lecture de fichier*/
DATA europe ;
INFILE 'europe.txt' ;
INPUT pays $ population superficie pib date monnaie $ ;
RUN ;
PROC PRINT;
RUN ;
```

- Créer à l'aide d'un éditeur de texte (BlocNote) le fichier de données « europe.txt » qui contiendra seulement la première ligne des données (la ligne avec l'Allemagne). Vous devrez modifier le programme pour indiquer le chemin absolu (depuis la racine C:/) de votre fichier. De plus vous ferez attention lors du nommage de votre fichier, en évitant en particulier de le nommer 'europe.txt.txt'.
- Taper et exécuter les instructions SAS ci-dessus. Analyser la fenêtre Journal et la fenêtre de résultats. Corriger vos instructions si nécessaire pour obtenir une exécution correcte.

8 Sauvegarde des données dans un fichier binaire SAS :

La liste d'instructions suivantes à pour but de sauvegarder les données étudiées dans un fichier binaire SAS. Les fichiers binaires SAS permettent de sauvegarder une information comprimée et sont d'accès plus rapide.

```
/*Analyse des données européennes, sauvegarde des données*/
LIBNAME repdata 'c:/temp';
DATA repdata.europe ;
        INFILE 'europe.txt' ;
        INPUT pays $ population superficie pib date monnaie $ ;
        RUN ;

PROC PRINT data=repdata.Europe;
RUN;
```

- Etudier et exécuter les instructions SAS ci-dessus. Analyser la fenêtre journal « .log » et la fenêtre sortie contenant les résultats.
- Vérifier que le fichier au format SAS a bien été crée dans le répertoire adéquat. Quel est son nom ?
- Sauvegarder votre programme.

9 Accès aux données stockées dans un fichier binaire SAS:

La liste d'instructions suivantes permet d'accéder aux données du fichier SAS. La procédure

CONTENTS produit dans le fichier résultat un résumé des caractéristiques du fichier DATA. Si l'option DATA n'est pas mentionnée dans l'instruction PROC CONTENTS, il s'agit du dernier fichier DATA crée, donc fic1.

```
/*Analyse des données européennes, accès données SAS*/
LIBNAME repdata 'c:/temp';
DATA fic1 ;
        SET repdata.europe;
RUN ;
```

```
PROC PRINT data=fic1 ;
RUN ;
```

```
PROC CONTENTS;
RUN ;
```

- Taper et exécuter les instructions SAS ci-dessus.
- Analyser la fenêtre journal « .log » et la fenêtre sortie contenant les résultats.

10 Ecrire dans le journal : PUT

La commande PUT permet d'écrire dans le journal, en particulier pour afficher la valeur des variables :

```
Data Tab;
Input X;
put x=;
CARDS;
8
6
;
RUN;
```

2 LECTURE ET PRÉSENTATION DES DONNÉES

Les données que nous avons lues étaient séparées par des blancs. Dans la pratique, les données peuvent contenir des blancs (comme un titre d'un film) ou bien être accolées les unes aux autres.

1 Lecture de données organisées en colonnes :

Les observations du tableau suivant contiennent le nom d'un pays (New Zealand), puis trois valeurs numériques :

```
NEW ZEALAND 1950 1914 80.9
NEW ZEALAND 1951 1960 72.5
```

Le blanc dans le nom du pays gêne pour la lecture des données. Comme ces données sont organisées en colonne, nous pouvons donner explicitement la position de la première variable comme suit :

```
data desc;
input pays $ 1-12 an va vb;
cards;
```

Le « 1-12 » indique que la variable pays sera lue du caractère 1 au caractère numéro 12. Les espaces ne servent plus alors à délimiter cette variable. Cette solution vient avec une contrainte : pour chaque ligne de données, il ne peut y avoir que la variable pays sur les caractères de 1 à 12.

- Taper et exécuter les instructions SAS ci-dessus pour créer le tableau *desc*, puis utiliser la procédure PROC PRINT. Pour cette question je donne le programme complet :

```
data desc;
input pays $ 1-12 an va vb;
cards;
NEW ZEALAND 1950 1914 80.9
NEW ZEALAND 1951 1960 72.5
;
proc print; run;
```

- Spécifier également la position de la variable **an**, par exemple avec an 13-16.
- Ajouter une ligne avec « France », en prenant garde à ne pas mettre d'autres données que le nom du pays entre les positions 1 et 12.

2 Spécifier le début de la variable

Nous avons les données suivantes :

```
Belgique "données à vérifier"          166 23
France "l'année 1982 est manquante"  191 21
```

Il y a des commentaires que nous ne souhaitons pas lire entre la première et la deuxième variable. Notez bien que le 166 est aligné avec le 191. La solution pour lire ces données consiste à spécifier le début de la deuxième variable à l'aide de @ :

```
Input pays $ @37 X Y;
```

L'instruction @37 demande à SAS de placer le curseur permettant de lire les données au caractère 37. L'éditeur de SAS indique en bas à droite la colonne sur laquelle est placé le curseur.
A nouveau, je donne le programme complet :

```
Data tab ; input pays $ @37 X Y; cards ;
Belgique "données à vérifier"          166 23
France "l'année 1982 est manquante"  191 21
;
proc print ; run ;
```

3 Lecture de données à longueur supérieure à 8

Considérons les données suivantes :

```
Irlande Femmes 598 Services 79.3
Irlande Hommes 931 Agriculture 15.6
Luxembourg Femmes 56 Industrie 6.6
Luxembourg Femmes 56 Services 92.1
```

En essayant de lire ces données, nous nous apercevons qu'elles sont tronquées à partir de 8

caractères. Ceci est le comportement automatique de SAS. Si ces données étaient organisées en colonne, nous pourrions utiliser la méthode précédente. Ici nous utilisons la syntaxe suivante :

```
input pays $ :12. genre $ nb typ $ val;
```

La commande « :12. » indique que la variable pays peut contenir jusqu'à 12 caractères.

➢Ecrire le programme permettant de rentrer les données précédentes. Il faudra en particulier veiller à ne pas tronquer « agriculture ».

4 Utiliser les formats pour lire les données

Lorsque nous connaissons la longueur des données d'une variable, nous pouvons le préciser à l'aide des formats. Ceci permet de lire des fichiers dans lesquels des données sont accolées. Voici par exemple les températures, degrés d'hygrométries et superficie pour les différentes pièces d'un appartement :

```
Salon 2041186
Cuisine 2146115
Chambre 1943108
```

La température est donnée sur 2 chiffres, le degré d'hygrométrie également et la superficie est donnée avec 3 chiffres et sans la virgule. Il faut ainsi lire que le salon a une superficie de 18,6 m^2.

La syntaxe est la suivante :

```
input nom $ temp 2. hydr 2. aire 3.1;
```

Le format « 2. » indique que la variable **temp** est entrée sur 2 chiffres. Le format « 3.1 » indique que la variable **aire** est entrée sur trois chiffres et qu'il faut placer une virgule avant le dernier chiffre.

➢Ecrire le programme permettant de rentrer les données précédentes.

5 Des données sur plusieurs lignes

SAS par défaut va à la ligne lorsque l'observation courante est incomplète pour pouvoir lire les autres variables. Ainsi les deux programmes suivants donnent le même tableau résultant :

```
Data Tab;
Input x y;
cards;
8 6
;
```

```
Data Tab;
Input x y;
cards;
8
6
;
```

6 Comportement du INPUT

Soit le programme

```
Data Tab ; input x ; cards ;
2  4  6
13  11
; proc print ; run ;
```

Vous observerez que le tableau ne contient que deux observations, avec les valeurs 2 et 13. Lorsque le INPUT a fini de lire ses variables, il va automatiquement à la ligne. Ce comportement peut être modifié comme indiqué dans le point suivant.

7 Lire plusieurs observations par ligne

Jusqu'à présent les observations ont été entrées sur des lignes séparées. Nous pouvons avoir des notes sur 20 à deux examens présentées comme suit :

```
12 15 16 08 11 09
19 18 06 04
```

Pour pouvoir lire plusieurs observations par ligne, il faut utiliser l'instruction @@ :

```
input note1 note2 @@;
```

- Ecrire le programme permettant de rentrer les données précédentes dans le data *notes*.
- Ajouter la variable note3.
- Enlever le @@ et regarder le résultat.

Il faut mettre @@ quel que soit le nombre de variable. En particulier l'instruction @@@ n'existe pas.

8 Mettre des labels à la place des noms de variables

Les noms de variables ne peuvent contenir d'espaces et ont 8 caractères maximum. Pour avoir des noms plus précis, on utilise des labels. Exemple :

```
proc print data=notes label;
label note1='note du premier examen' note2='note
du deuxième examen';
run;
```

Le premier « label » indique que la procédure PRINT doit remplacer les noms des variables par leurs labels si ceux-ci existent. Le second « label » indique les labels de **note1** et de **note2**.

9 Ajouter des labels à un tableau

Il est possible de préciser les labels lors de l'étape DATA par la même instruction. Ces labels sont alors permanents. Pour associer les labels à un tableau tab, il faut les définir lors de l'étape Data. Dans le programme suivant, on ajoute ces labels, puis on affiche le tableau avec les labels :

```
Data notes; set notes;
label note1='note du premier examen' note2='note du deuxième examen';
proc print label; run ;
```

10 Utiliser un format pour l'écriture.

Nous étudions des individus dont le genre est codé par « 1 » pour masculin et « 2 » pour féminin. Lors de l'écriture du tableau, nous aimerions voir apparaître ces mots à la place de « 1 » et « 2 ». Nous utilisons pour ce faire la procédure FORMAT :

```
proc format ;
value genref 1='masculin' 2='feminin';
run;
data donnees;
input genre @@;
cards;
1 2 2 2 1 2
;
run;
proc print;
format genre genref.;
run;
```

Après avoir défini le format genref, nous l'utilisons dans la procédure PRINT. Remarquer le point après genref lors de l'utilisation du format.
➢Taper le programme précédent.

11 Cas particulier du format alphanumérique

Pour utiliser des caractères à la place des chiffres pour un format, il faut utiliser le signe $ lors de la définition et de l'utilisation du format.
Voici un petit programme complet :

```
Data tab;
input x $ @@ ;
cards;
o n n o
;
proc format;
value $ rep o='oui' n='non';
proc print;
format x $rep.; run;
```

12 Trier un tableau

Pour un groupe d'enfants, nous connaissons le prénom, l'âge, le genre, la taille, le poids et la couleur des cheveux (1= brun , 2= châtain et 3= blond). Ce tableau est sur ma page web.
➢Aller sur ma page web(nom de famille + nanterre), demander à ouvrir le tableau enfants avec SAS
➢Fermer le tableau enfants, aller dans la bibliothèque « tmp(n+1) » et copier coller le tableau vers la bibliothèque Work
➢Renommer le tableau au besoin.
Une fois le tableau **enfants** présent dans 'Work'**,** nous trions les observations suivant la taille :

```
proc sort data=enfants;
by taille;
run;
```

- Faites un PROC PRINT sur le tableau enfants, en utilisant un format pour les cheveux.
- Trier les données suivant le poids, puis la couleur des cheveux. Trier suivant la couleur des cheveux *et* la taille. La syntaxe pour trier suivant deux variables est par exemple :

```
by cheveux taille;
```

13 Enregistrer les observations triées dans un tableau

Le programme précédent permet de trier le tableau enfants. Ici nous allons trier les observations suivant l'âge et enregistrer le résultat dans un nouveau tableau :

```
Proc sort data=enfants out=enfants_age;
by age;
Proc Print data=enfants_age;
RUN;
```

14 Sélectionner les observations à trier

Nous trions les observations selon le poids, en ne considérant que les garçons :

```
Proc sort data=enfants (WHERE=(genre='M')) out=garcons_poids;
by poids;
RUN;
```

15 La commande BY.

La commande BY permet de répéter la procédure pour chaque modalités de la variables spécifiée. Le tableau doit être trié suivant cette variable. Exemple :

```
Proc sort data=enfants;
by genre;
proc print data=enfants;
by genre;
run;
```

➢ Entrer le programme précédent.
➢Faire « by cheveux ».

16 Histogrammes et autres graphiques avec GCHART

Pour tracer l'histogramme des poids du tableau **enfants**, nous entrons :

```
proc gchart data=enfants;
vbar poids;run;
```

Pour les variables discrètes comme la couleur des cheveux, nous devons coder :

```
pie cheveux/discrete value=inside slice=outside;
```

Nous avons placer des options pour le graphique après le signe « / ».

- Entrer les programmes précédents.
- Essayer différents graphiques (vbar, hbar, pie) sur les différentes variables.
- Pour combiner GCHART avec BY genre, nous entrons :

```
proc gchart data=enfants;
vbar poids; by genre; run;
```

Il faut que le tableau soit trié par rapport au genre avant d'exécuter cette étape.

17 Tracé de fonctions avec GPLOT

Nous souhaitons voir l'évolution du poids par rapport à la taille. Nous entrons

```
proc gplot data=enfants;
plot poids*taille;run;
```

➢ Entrer le programme précédent.
➢Tracer l'évolution de la taille en fonction de l'âge.

3 MANIPULATION DES DONNÉES

Manipulation sur les observations

1 Création de petits tableaux d'étude

Créer les tableaux suivants :

Tab1	Tab2	Tab3
X Y	X Z	X Y
1 2	3 5	2 3
7 9	6 8	

Par exemple pour le premier tableau :

```
Data Tab1 ; input X Y ;
cards ;
1 2
7 9
;run ;
```

2 Concaténation de tableaux avec SET

Pour concaténer deux tableaux, nous entrons :

```
data tab4;
set tab1 tab2;
run;
```

Le nouveau tableau contient toutes les observations et toutes les variables des tableaux précédents. Si une variable n'existait pas dans un tableau, les valeurs sont notées manquantes par un point.

- Faire la concaténation de deux tableaux.
- Concaténer trois tableaux.

3 Proc APPEND

Pour ajouter des observations à un tableau, il est plus indiqué d'utiliser Proc Append :

```
Proc APPEND base=tab3 data=tab1;
proc print data=tab3 ;
run;
```

Ceci ajoute les observations de tab1 dans tab3. Tab3 doit posséder toutes les variables de Tab1. Notez que pour l'affichage l'utilisation de l'option data=tab3, car tab3 n'est pas considéré comme le dernier tableau créé.

4 Sélection des observations avec WHERE

A partir du tableau **enfants** de la séance précédente, nous aimerions avoir un tableau avec seulement les enfants bruns. Nous sélectionnons ces observations avec l'instruction SET :

```
data bruns;
set enfants;
where cheveux=1;
run;
```

- Entrer le programme précédent.
- Créer le tableau des garçons avec *where genre= «M »*.
- Sélectionner les enfants faisant plus de 1m60
- Essayer *where cheveux=1 and taille<150 ;*

L'instruction WHERE est un filtre, qui ne laisse passer que les observations vérifiant une certaine condition.

5 Suppression d'observations avec IF et DELETE

Inversement, nous pouvons choisir de supprimer des observations :

```
data filles;
set enfants;
if genre="M" then delete;
run;
```

- Entrer le programme précédent.

6 Sélection des observations avec IF et OUTPUT

Si nous préférons spécifier les observations que nous souhaitons garder avec IF, nous utilisons la commande OUTPUT à la place de DELETE. La commande OUTPUT écrit l'observation courante dans le tableau que nous sommes en train de créer.

```
data filles;
set enfants;
if genre="F" then output;
run;
```

La commande OUTPUT était en réalité toujours sous-entendue dans les programmes précédent. Lorsque nous spécifions au moins une fois la commande OUTPUT dans notre étape DATA, SAS ne la rajoute plus par défaut à la fin de cette étape et donc les observations correspondantes aux garçons sont bien mises de côté dans ce programme.

7 IF ou WHERE ?

La commande IF permet de faire ce que l'on peut faire avec la commande WHERE. La commande WHERE, elle, agit comme un filtre sur les observations et est beaucoup plus rapide que IF. A chaque fois que cela est possible, il faut utiliser WHERE, et n'utiliser IF que lorsque cela est nécessaire.

8 Créations de plusieurs tableaux avec IF et OUTPUT

Nous aimerions créer en une seule étape DATA un tableau contenant les garçons et un tableau pour les enfants châtain :

```
data garcons chatains;
set enfants;
if genre="M" then output garcons;
if cheveux=2 then output chatains;
run;
proc print data=garcons ;
proc print data=chatains ; run ;
```

- Entrer le programme précédent.
- Créer d'autres tableaux

Vous remarquerez qu'il est nécessaire de faire un proc print par tableau, et que cette fois les noms des tableaux sont indiqués dans les proc print.

9 Séparation des observations avec SELECT, WHEN et OUTPUT

Pour découper un tableau en fonction des modalités d'une variable, nous utilisons l'instruction SELECT :

```
data garcons filles;
set enfants;
select(genre);
when("F") output filles;
otherwise output garcons;
end;
run;
```

- Entrer le programme précédent.

10 Syntaxe générale pour SELECT

La commande Select a deux syntaxes. Celle ci-dessus, la syntaxe raccourcie, permet de séparer un tableau suivant les modalités d'une variable. Pour utiliser d'autres conditions, nous utilisons la syntaxe générale :

```
Select;
when(condition1) …;
when(condition2) …;
otherwise …;
end;
```

Pour les conditions, on écrira par exemple *when(taille<140)*.
Si on veut séparer les observations du tableau **enfants** en trois groupes suivant leur taille:

```
data petits moyens grands;
set enfants;
select;
when(taille<150) output petits;
when(taille<160) output moyens;
otherwise output grands; end;run ;
```

- Afficher les trois tableaux *petits*, *moyens* et *grands* après les avoir créés.

11 Combiner WHERE et SELECT

Pour créer les tableaux des filles et des garçons de moins d'1m50 :

```
data garcons filles;
set enfants; where taille<150 ;
select(genre);
```

```
when("F") output filles;
otherwise output garcons;
end;
run;
proc print data=garcons ; proc print data=filles;run ;
```

Manipulation sur les variables

12 Sélection des variables avec KEEP

Parfois le tableau contient plus de variables qu'il n'est nécessaire. Nous gardons celles qui nous intéressent avec l'instruction KEEP :

```
data nomgenre;
set enfants;
keep prenom genre;
run;
```

- Entrer le programme précédent.
- Créer d'autres tableaux en sélectionnant différentes variables.

13 Utiliser KEEP dans une procédure PRINT

```
Proc print data=enfants (keep = prenom taille);
run;
```

- Cette fois l'option data= est obligatoire, car *KEEP* est une option du tableau et non d'une option du proc print.

14 Suppression de variables avec DROP

L'instruction DROP est l'inverse de l'instruction KEEP, et indique la ou les variables à supprimer.

- Dans les programmes précédents, remplacer KEEP par DROP.

15 Renommer les variables

On dispose d'un tableau contenant une unique variable appelée X. Pour renommer X en Y, on utilise l'instruction RENAME :

```
Data Tab ; x=3 ; Proc Print ;
Data Tab;
set Tab;
rename x=y; Proc Print ; run ;
```

Cet exemple suppose que Tab a la variable *x* et pas la variable *y*.

Fusion de tableaux

16 Fusion de tableaux avec MERGE

L'instruction MERGE opère une fusion horizontale des tableaux :

```
data fusion;
merge garcons tab2;
run;
```

Si une variable est présente dans les deux tableaux, elle prend la valeur donnée par le dernier

tableau. Cet exemple suppose que vous avez créé les tableaux *garcons* et *tab2* dans cette session.

- Fusionner différents tableaux.

17 Fusion de tableaux avec MERGE+BY

Imaginons que sur notre groupe d'enfants, nous possédions plusieurs tableaux : un pour le poids, l'autre pour la taille et un pour le genre :

```
Data poids;
set enfants;
keep prenom poids;
proc sort data=poids ; by prenom ;
data taille; ...
```

Pour réunir toutes ces informations dans un seul tableau et en supposant que les enfants ont des noms différents, nous utilisons l'instruction MERGE+BY nom :

```
data fusion_by;
merge poids taille genre;
by prenom;
run;
```

!! Les tableaux doivent être préalablement triés suivant la variable **prenom**.

➢ Séparer le tableau **enfants** en trois tableaux avec des instructions KEEP.

➢ Réaliser la fusion avec MERGE+BY.

18 L'instruction IN avec MERGE+BY

Créez les deux tableaux FA et FB suivants :

T	X	Z
4	X4	Z4
5	X5	Z5
6	X6	Z6

T	Y	Z
5	Y5	ZP5
6	Y6	ZP6
7	Y7	ZP7

Pour la création de FA, le input sera donc :

```
input T X $ Z $ ;
```

Ces tableaux possèdent tous deux des données pour T=5 et T=6. Pour réaliser l'intersection sur la variable T de ces deux tableaux, nous entrons :

```
Data tab; merge FA(in=A) FB(in=B); by t;
if A and B;
run;
```

Les instruction *in=* créent deux variables temporaires *A* at *B.* La variable *A* indique si la modalité courante de *T* appartient au tableau FA.

- Créer FA et FB, en faisant attention aux variables alphanumériques, puis entrer le programme précédent.
- Réaliser la concaténation de FA et FB à l'aide de SET.
- Réaliser la fusion de FA et FB par rapport à *T* à l'aide de MERGE+BY.
- Essayer « if not B » à la place du « if A and B ».
- Essayer « if A »

19 Un problème de transformation des données

On dispose de plusieurs valeurs de températures pour deux stations, sous la forme

```
Sta1 20 18 15.4
Sta2 17 15 9
```

A l'aide de l'instruction OUTPUT et de variables intermédiaires, on va écire le programme permettant de lire ces données (avec une seule température par observation). On doit donc obtenir comme tableau :

```
Station  temp
sta1     20
sta1     18
sta1     15.4
sta2     17
sta2     15
sta2     9
```

La solution est :

```
Data meteo ;
input sta $ t1-t3 ;
t=t1 ; output ; t=t2 ; output ; t=t3 ; output ;
cards ;
sta1 20 18 15.4
…
;
```

Une fois le programme compris, vous pouvez ajouter un *keep* pour ne garder que les variables d'intérêt, c'est à dire *sta* et *t*.

4 OPÉRATIONS, FONCTIONS, BOUCLES

1 Opérations élémentaires

On peut calculer de nouvelles variables à l'aide des variables existantes :

```
Data nombres; Input x y @@;cards;
5 5 2 -3 -4 7.12 8 1.6 2 0
;run;
```

```
Data calculs;
set nombres;
a=x+y; b=x-y; c=x**y; d=min(x,y); e=max(x,y);
f=x*y; g=x/y;
run;
```

La notation x**y signifie « x à la puissance y ».

2 Fonctions courantes

```
Data fcts; set nombres;
a=abs(x); b=exp(x); c=int(y); d=log(y);
e=log10(y); f=sign(y); g=sqrt(x);
run;
```

La fonction *log()* retourne le logarithme népérien, la fonction *log10()* retourne elle le logarithme à base 10.

3 Les boucles

On veut créer un tableau donnant le logarithme des nombres entre 0 et 1 par pas de 0,01. On automatise la création de ce tableau à l'aide des commandes DO, TO, BY et END ainsi que OUTPUT:

```
Data tab; DO x=0.01 TO 1 BY 0.01;
y=log(x);
OUTPUT tab;END;run;
```

➢Créons maintenant un tableau de 10 données aléatoires X à l'aide de l'instruction RAND('binomial',0.4,20) :

```
Data tab; DO i=1 TO 10;
x=rand('binomial', 0.4, 20);
OUTPUT;END;run;
```

4 Comparaison Proc Append et SET

Le programme suivant permet de constater la différence de vitesse d'exécution entre un Proc APPEND et un SET. Vous devrez regarder dans le journal les temps d'exécution de la commande SET et de PROC APPEND.

!!! Ne faites pas de PROC PRINT.

```
data tab;
do i=1 to 2000000; output; end;
data tab2;
do i=1 to 1000; output; end;
data tab3;
set tab tab2;
run;
proc append base=tab data=tab2;
run;
data tab;data tab3; run;
```

5 Les compteurs

Il existe une méthode simple pour implémenter un compteur sous SAS :

```
data tab;
input x @@;
y+1;
z+x;
cards;
2 1 3 2
;
```

La variable Y est un compteur incrémenté de 1 à chaque fois. La variable Z est un compteur incrémenté de X à chaque fois.

6 Les cumuls : l'instruction RETAIN

Entrez le programme suivant :

```
data tab;
input x @@;
retain y 0;
y=y+x;
cards;
1 3 6 7 9
;
```

Vous pouvez observer que la variable y contient les sommes successives de la variable x.

L'instruction RETAIN permet de retenir la valeur d'une variable d'une observation à une autre. Ainsi

```
retain x y 3;
```

indique de garder en mémoire d'une ligne sur l'autre les valeurs des variables x et y, la variable y étant initialisée à 3.

Fonctions statistiques

7 Probbnml

L'appel Probbnml(p,n,m) retourne la probabilité qu'une binomiale de paramètre n et p soit inférieure ou égale à m. Pour donner la probabilité qu'une binomiale B(60,0.8) soit inférieure ou égale à 32 :

```
Data tab ;
x=probbnml(0.8,60,32);run ;
```

- Pour avoir la probabilité qu'une binomiale B(60,0.8) soit supérieure ou égale à 57 :

```
Data tab ;
x=1-probbnml(0.8,60,56);run ;
```

- A l'aide d'une boucle, créer un tableau contenant les probabilité qu'une binomiale B(20,0.6) soit inférieure à i pour i variant de 0 à 20 (ce sont les valeurs extrêmes de la binomiale considérée).

8 Probnorm

L'appel Probnorm(x) retourne la probabilité que la gaussienne centrée réduite soit inférieure à x.

- Pour avoir la quantité P(X<1,02) pour X normale :

```
Data tab ;
a=probnorm(1.02);run ;
```

- Donner P(X<x) pour x variant de 4 à 6 par pas de 0,025

9 Probt

L'appel Probt(x,df) retourne la probabilité qu'une variable de Student de paramètre df soit inférieure ou égale à x

- Donner P(T<1,4) pour T de loi de Student de paramètre 4
- Donner P(T<1) pour T loi de Student de paramètre i variant de 5 à 20.

10 Probit

Probit(p) retourne le quantile associé à p pour la loi normale, c'est à dire qu'il retourne la valeur x telle que P(X<x)=p pour X de loi normale.

- Trouver x tel que P(X<x)=0,98.

Fonctions aléatoires

11 Ranuni

L'appel Ranuni(seed) retourne un nombre pseudo-alétoire suivant la loi uniforme sur [0,1]. Seed signifie graîne en anglais. C'est le premier terme de la suite pseudo-aléatoire qui permet d'obtenir nos nombres aléatoires. Il suffit de la remplacer par un entier, par exemple ranuni(5).

- Faire un tirage de 30 nombres et les afficher avec un histogramme :

```
Proc gchart ; vbar x;run ;
```

12 Rannor

L'appel Rannor(seed) retourne un nombre pseudo-alétoire suivant la loi normale.

- Faire un tirage de 50 nombres et les afficher avec un histogramme

5 LECTURE ET PRÉSENTATION DES DONNÉES II

1 Options de Infile : DLM, Firstobs

Parfois un fichier ressemble à ceci :

```
X;Y;Z
56;12;3
13;9;18
```

On remarque donc qu'il ne faut pas lire la première ligne, et que le caractère de séparation des données n'est pas l'espace mais le point-virgule. La solution est de spécifier des options pour le infile :

```
Data tab;
infile 'c:\temp\essai.txt' DLM=';' Firstobs=2;
input x y z;
```

L'option DLM permet de spécifier le caractère de délimitation, l'option Firstobs permet de spécifier la première ligne à lire du fichier.

2 Infile et Cards

Nous pouvons profiter des options disponibles pour la commande INFILE tout en utilisant la commande CARDS. Essayez le programme suivant :

```
data tab;
infile cards dlm=',';
input x @@;
cards;
8,6,7
9,12,3
;
```

3 Cards4

Le caractère ';' est un caractère spécial pour CARDS et ne peut être utilisé dans les données. Pour résoudre ce problème, il est possible d'utiliser la commande CARDS4, qui utilise quatre points-virgules accolés pour indiquer la fin des données. Essayez le programme suivant :

```
data tab;
infile cards dlm=';';
input x @@;
cards4;
8;6;17
9;12;3
;;;;
```

4 Nobs et End

Lors de l'instruction SET, les options Nobs= et End= permettent de créer des variables temporaires. Celle avec Nobs contient le nombre d'observations du tableau, tandis qu'avec End nous avons une variable booléenne indiquant si nous nous trouvons à la dernière ligne du tableau. Exemple :

```
Data Tab2;
set Tab nobs=counter end=last_obs;
nombre=counter;
keep nombre;
if last_obs then output;
```

Ce programme crée un tableau Tab2 contenant une ligne et une variable. La valeur unique de ce tableau est le nombre d'observations dans le tableau Tab.

5 Point

Il est possible de spécifier le numéro de l'observation qui nous intéresse lors de l'instruction SET à l'aide de l'option Point :

```
data tab2;
do obsnum=1 to num by 2;
set tab nobs=num point=obsnum;
output;
end;
stop;
```

Remarquez le output, nécessaire pour écrire l'observation dans le tableau Tab2 lorsque l'on spécifie l'option Point, ainsi que l'instruction STOP également nécessaire en ce cas.

L'option Point s'utilise obligatoirement avec une variable et non simplement un nombre.

1.Pour lire à l'aide de Point la 3ème observation d'un tableau Tab :

```
data tab3;
a=3 ;
set tab point=a;
output; stop;
```

2.En une seule étape Data, lire la 2ème observation d'un tableau Tab et la 3ème d'un tableau Tab2. Le programme contiendra en particulier :

```
data tab4;
a=2 ;
set tab point=a;
output; a=3 ; set tab2 point=a ; ...
```

3.Ecrire le programme permettant de renverser un tableau.

6 Datalines

Vous rencontrerez souvent DATALINES dans les programmes SAS. DATALINES est un synonyme de CARDS, de même DATALINES4 est un synonyme de CARDS4.

7 Plusieurs INPUTs avec @

Le caractère @ à la fin d'une instruction INPUT indique qu'il ne faut pas changer de place le curseur et ne pas passer à l'observation suivante lors d'une instruction INPUT ultérieure. Ainsi

```
data donnees;
input x @;
input y;
cards;
8 9
2 4
;
```

donne le tableau espéré, ce qui n'est pas le cas sans @.

- Entrer le programme précédent. Ajouter une troisième variable.

8 Retour sur l'exemple météo

Dans ce travail de la troisième feuille, trois températures étaient associées à une station. Pour lire ces données on utilisait des variables intermédiaires. Nous présentons une solution plus élégante à l'aide de @ :

```
data meteo;
input station $ @;
do i=1 to 3;
input temp @; output; end;
keep station temp;
cards;
sta1 20 18 15.4
sta2 17 15 9
;
```

- Entrer le programme précédent.

Maintenant le nombre de températures est variable et est indiqué juste après la station :

```
Sta1 3 17 16.5 8
Sta2 2 10.5 14
```

Ainsi le 3 signifie qu'il y a 3 températures.

- Ecrire le programme permettant de lire ces données. Le premier input deviendra :

```
input station $ n @;
```

9 Suite de variables

Pour lire quatre variables nommées X1, X2, X3 et X4, on utilise lors du Input la notation X1-X4 :

```
Data tab ; Input X1-X4; cards ;
1 5 8 2
;
```

Pour affecter un même format dans le Input à plusieurs variables, on utilise des parenthèses :

```
Input (X1-X4 Y) (2.);
```

10 Array

Il est parfois pratique de ranger les variables dans un tableau. Par exemple lorsque la valeur de la variable Xi dépend de son indice i :

```
Data Tab;
Array c{4} X1-X4;
do j=1 to 20; do i=1 to 4;
c{i}=cos(i+j);
end;output;end;
```

Les informats

11 Les informats

Un informat est un format d'entrée pour une variable, spécifié lors de l'instruction INPUT. C'est en qulque sorte l'inverse du format pour l'affichage. Par exemple pour le genre des enfants sur la feuille 2, on entrait 1 ou 2 dans le tableau mais on souhaitait avoir 'M' ou 'F' lors de l'impression. Avec un informat on peut entrer le genre directement avec 'M' et 'F' tout en ayant 1 ou 2 (les variables numériques sont souvent plus commodes) dans le tableau.

12 Définir un informat avec invalue

La syntaxe ressemble à la création d'un format pour l'affichage :

```
proc format;
invalue grade 'A'=4 'B'=3 'C'=2 'D'=1 'F'=0;
```

On peut maintenant lire les données suivantes en récupérant un tableau avec des données numériques :

```
Data tab;
input prenom $ (note1-note3) (:grade.);
cards;
Frank   A B F
Samuel  A B C
Sara    A C B
Elise   B . A
```

- Entrer les données précédentes
- Calculer la moyenne de chaque étudiant à l'aide de l'instruction

```
z=mean(of note1-note3);
```

que l'on placera juste avant le CARDS.

13 Lire la virgule française avec un informat

La virgule de SAS est le point. Pour lire la virgule française on utilise l'informat NUMX. :

```
data virgule;
input x numx.;
cards;
4,8
6,31
;
```

14 Attacher un format à une variable dans l'étape DATA

Lors de l'étape DATA, on peut attacher un format à une variable. Ce format sera utilisé par défaut pour chaque affichage de cette variable. Voici un exemple :

```
data form;
input x @@;
format x 4.2; cards;
8 2.3 9.2 32 456 2 8
;
```

- Entrer cette étape DATA puis afficher le tableau à l'aide d'un simple proc print.
- Immaginons que ces chiffres soient des prix que l'on souhaite afficher avec les centimes. Mettre le format 7.2 pour avoir le bon affichage.

La commande PICTURE dans PROC FORMAT

15 Utilisation de picture

L'instruction PICTURE dans proc format est plus souple et plus puissante que l'instruction VALUE. On gère les zéros situés en début des nombres, les espaces et autres caractères entre les chiffres.

16 Exemple de picture

```
proc format;
   picture phonenum other='90 00 00 00 00';
```

Ce format va permettre d'afficher des numéros de téléphones :

```
data telephone;
input tel @@; cards;
0245703265 125643512 123
;
proc print;
format tel phonenum.; run;
```

- Entrer les instructions précédentes
- Changer le 9 en 0 dans l'instruction picture.
- Remplacer les 0 par des 9 dans l'instruction picture.

17 Les digit selectors

Ce sont les chiffres qui apparaissent dans l'instruction picture. Le chiffre 9 indique qu'il ne faut pas supprimer les zéros du début du nombre, alors que le chiffre 0 indique au programme qu'il doit les

enlever.

18 Les sélecteurs

On peut créer plusieurs cas pour appliquer différents formats aux nombres. L'instruction OTHER indique que le format qui suit va être appliqué dans tous les cas non encore considérés. Ainsi dans l'exemple précédent ce format est appliqué à tous les nombres. Voici un exemple où l'affichage dépend du nombre :

```
proc format;
picture resume low-140='petit' 141-155='moyen'
155-high='grand';
```

On crée un tableau contenant des taille que l'on affiche avec ce format :

```
data donnees;
input taille @@;
cards;
132 141 155 189 132 145 172
;
proc print;
format taille resume.;
run;
```

- Entrer les instructions précédentes.

19 L'option prefix dans picture

Dans l'exemple précédent il n'y a pas de digit selectors et les chaînes de caractères s'affichent telles quelles. S'il y a au moins un chiffre dans la chaîne de caractère, tous les caractères précédents dont oubliés. Pour pouvoir placer des caractères en tête de sortie, on dispose de l'option PREFIX.
Soit par exemple des numéros de téléphones sur dix chiffres. On souhaite les afficher en mettant automatiquement le préfix de la France (+33). La solution est la suivante :

```
proc format;
picture teleph other='09 99 99 99 99' (prefix='(+33)');
```

Le préfixe est placé tel quel en début du nombre.

- Entrer ce format puis l'utiliser dans un PROC PRINT.

20 Récapitulatif

Nous allons afficher la taille avant la classification pour les grands, et la taille après la classification pour les petits (pour les moyens nous affichons seulement la taille en incorporant un « m »). La solution est :

```
proc format;
picture resume low-140='000' (prefix='petit') 141-155='0m00' 155-high='000
grand';
```

21 L'option NOEDIT dans picture

On souhaite parfois afficher des chiffres sans qu'ils soient interprètés en tant que digit selectors. On utilise pour ce faire l'option NOEDIT :

```
proc format;
picture km 1-99='000000'
           100-high='>100 kilomètres' (noedit);
```

Avec NOEDIT, le nombre 100 dans '>100 kilomètres' ne sera pas interprèté mais rendu tel quel.
On utilise ce format avec les données suivantes :

```
data temp;
input nom $ distance; cards;
Marc 19
Anne 258
Christophe 600
Sophie 27
;
```

- Entrer les instructions précédentes. Imprimer les données à l'aide du format 'km'.
- Enlever l'option *noedit* et constater que les chiffres ont été considérés comme des digit selectors.

22 Utilisation de picture : changement d'unités

A l'aide de picture on peut multiplier les données lors de leur affichage, permettant ainsi des changements d'unité. Ainsi pour convertir en miles des données exprimées en pied nous entrons :

```
proc format;
   picture feet other='000000009' (mult=5280);
```

Puis

```
data feet;
   input miles @@;
   format miles feet.;
   cards;
1   1.5   2
;
```

6 LES DATES DANS SAS

1 Enregistrement des dates sous SAS

Le logiciel SAS enregistre les dates par un nombre. Ce nombre indique le nombre de jours qui se sont écoulés depuis le 1er janvier 1960. Le chiffre 0 correspond donc à cette date initiale. Pour pouvoir entrer les dates et les lire de manière commode, il existe des formats (et informats) spécifiques.

2 Un exemple simple pour comprendre le rapport entre nombre et date

Soit le programme :

```
Data tab; do i=1 to 36 by 2; j=i ; output; end;
Proc Print; format i date7.; run;
```

Avec ce programme, on voit que SAS n'a pas de type spécifique pour enregistrer les dates, mais que tout se joue à l'affichage (et à la lecture, voir le point suivant).

3 Les formats DATE7. Et DATE9.

Le format date7 correspond par exemple à 18may88, tandis que le format date9 correspond à 11mar1991. Soit les dates suivantes :

```
8jan89 7feb54 22mar60 15apr73 29may75 28jun00
5jul23 7aug17 2sep4 19oct81 30nov98 13dec61
```

- Entrer le programme permettant de les lire, puis de les afficher avec les années sur quatre chiffres. Pour pouvoir utiliser les formats de date avec @@, il faut placer le symbole ':' avant le format comme suit :

```
input date :date7. @@;
```

L'affichage sur 4 chiffres pour la date se fera avec

```
Proc print ; format date date9. ;
```

- A l'aide d'une boucle DO, nous écrivons un programme permettant de retrouver les abrévations pour les douze mois de l'année :

```
Data tab; do i=1 to 365 by 31; output; end;
Proc Print; format i date7.; run;
```

4 Les formats DDMMYY8. Et DDMMYY10.

Ces formats permettent de lire des dates telles que 15/10/81 et 25/12/2005 respectivement. Soit les dates suivantes :

```
01/12/25 13/8/3 31/3/88 24/11/73
```

- Entrer le programme permettant de les lire, puis de les afficher avec les années sur quatre chiffres.
- Pour pouvoir utiliser les formats de date avec @@, il faut placer le symbole ':' avant le format.

5 Définition d'une période de cent ans pour les dates.

La période de cent ans couverte par les formats ayant deux chiffres pour les années peut être ajustée. L'option qui aide à déterminer est YEARCUTOFF=. Par défaut la valeur de cette option est 1920. Cela signifie que toute date contenant une année sur deux chiffres est considérée comme étant entre 1920 et 2019.

- Définir de nouvelles valeurs pour YEARCUTOFF et les tester sur les sections précédentes. Par exemple :

```
Option yearcutoff=1750;
Data tab; input x :date7.;cards;
8jan26
;
proc print; format x date9.; run;
```

6 Affectation directe d'une date

Pour affecter une date SAS à une variable à partir d'une date usuelle, on utilise par exemple :

```
x='15may1982'd;
```

Des fonctions pour les dates

7 Date, time et datetime

Les dates sont enregistrées dans SAS comme le nombre de jour s'étant écoulé depuis le 1er janvier 1960, le moment de la journée (time) est le nombre de secondes écoulées depuis minuit, et le datetime est le nombre de secondes écoulées depuis le 1er janvier 1960 à minuit.

8 Liste de fonctions

Voici une liste de fonctions sur les dates, que vous êtes invités à parcourir rapidement, mais sur lesquelles vous ne serez pas interrogés.

DATE()	Retourne la date courante
DATEJUL(arg)	Convertit une date sous format Julien en date SAS. Par exemple 1999364 est le 30 décembre 1999 en format Julien, et l'on écrira x=datejul(1999364).
DATEPART(arg)	Extrait la date d'une valeur Datetime
DATETIME()	Retourne le Datetime de l'instant présent
DAY(arg)	Retourne le jour du mois à partir d'une date SAS
DHMS(date, heure, min, sec)	Retourne un Datetime à partir de la date, heure, minute et seconde
HMS(heure, min, sec)	Retourne le Time à partir des heure, minute et seconde
HOUR(arg)	Retourne l'heure à partir d'une valeur Datetime ou Time.
INTCK('interval', from, to)	Retourne le nombre d'intervalles entre deux temps
MDY(mois, jour, année)	Retourne une date SAS à partir des mois, jour, année
MINUTE(arg)	Retourne les minutes à partir d'une valeur Datetime ou Time
MONTH(arg)	Reourne une valeur numérique de 1 à 12 pour le mois à partir d'une date ou d'un Datetime
QTR(arg)	Retourne la valeur numérique du trimestre à partir d'une date SAS ou d'un Datetime
SECOND(arg)	Retourne la valeur numérique de la partie seconde de 0 à 59 à partir d'un temps SAS
TIME()	Retourne le temps présent
TIMEPART()	Extrait la partie Time d'une valeur Datetime
TODAY()	Retourne la date courante
WEEKDAY(arg)	Retourne la valeur numérique du jour de la semaine à partir d'une Date ou d'un Datetime
YEAR(arg)	Retourne une valeur numérique sur 4 chiffres à partir d'une date SAS
YRDIF(startdate, enddate, basis)	Retourne la différence en années entre deux dates SAS
YYQ(year, quarter)	Retourne une date SAS en fonction de l'année et du trimestre.

7 VALEURS MANQUANTES

1 Introduction

Les observations d'un tableau SAS ne contiennent pas nécessairement une valeur pour chaque variable. Pour indiquer une valeur numérique manquante, il faut utiliser le point :

```
data tab;
input x y;
cards;
8 .
```

```
. 9
6
.
;
```

- Entrer ce programme
- Remarquer lors de l'affichage que SAS utilise le point pour indiquer les valeurs manquantes.

2 MISSOVER

Par défaut, SAS va à la ligne suivante si toutes les variables du Input n'ont pas été renseignée (comme dans le programme précédent pour la dernière observation).

Pour changer ce comportement et empêcher SAS de lire les observations sur plusieurs lignes, nous pouvons utiliser l'option MISSOVER :

```
data tab;
infile cards missover;
input x y;
cards;
8

9
6
;
```

Ainsi les dernières variables non renseignées prennent la valeur vide.

- Entrer le programme précédent
- Mettre un $ après le 'y'. Remarquer comment SAS gère les valeurs alphanumériques vides.

3 TRUNCOVER

Les lignes de données ont par défaut une longueur de 80, et une longueur de la taille de la ligne si la ligne est effectivement plus longue. Imaginons que nous souhaitions lire les 90 premiers caractères. Essayons avec :

```
data tab;
input x $ 1-90;
cards;
pp
jj
ii
mm
;
```

Entrez le programme précédent et remarquez que seulement une ligne sur deux est prise en compte.

Pour exécuter un INPUT, SAS place la ligne de données dans une variable temporaire nommée _INFILE_. Cette variable a une longueur de 80 (plus précisément, sa longueur est le maximum de 80, de sa longueur précédente et de la longueur de la ligne actuelle). Donc ici nous essayons de lire 90 caractères dans la variable _INFILE_ qui n'en compte que 80. Il ne s'agit pas de comprendre pourquoi SAS va finalement lire une ligne sur deux, mais juste de comprendre qu'il y a un problème de lecture dans le cas considéré.

Pour contourner le problème, nous utilisons l'option TRUNCOVER, qui permet de tronquer la lecture des lignes si celle-ci dépasse la longueur de la ligne :

```
data tab;
```

```
infile cards truncover;
input x $ 1-90;
cards;
pp
jj
ii
mm
;
```

- Entrer le programme précédent

4 Données manquantes spécifiées par l'utilisateur

Considérons le cas suivant : une variable X prend des valeurs numériques, mais nous ne disposons pas toujours de ces valeurs, et ceci pour différentes raisons. Par exemple cette valeur n'existe pas dans la base de donnée, ou bien pour des raisons de protection de données nous ne souhaitons pas la rendre disponible. Nous notons A et P ces deux types de valeurs manquantes.

Pour pouvoir spécifier ces valeurs manquantes spécifiques, nous utilisons l'instruction MISSING :

```
missing A P;
data tab;
input x @@;
cards;
3 A P C -8
;
```

Remarquez le missing en dehors de l'étape DATA. Une fois cette commande validée, elle le reste pour toute la session SAS (comme l'option formdlim). Effacer par la suite la ligne du "missing" n'a donc aucune influence.
La valeur « C » a été considérée comme manquante avec le simple point, car C n'était pas déclarée dans la liste des données manquantes spéciales.

5 Update 1

L'instruction UPDATE permet de mettre à jour un tableau à l'aide d'un second tableau, ces deux tableaux disposant d'une variable commune que l'on utilise comme clé :

```
data taba;
input x y t;
cards ;
1 3 8
2 4 6
;
data tabb;
input x z t;
cards;
1 5 1
2 3 1
;
data tab;
update taba tabb;
by x;
```

- Entrer le programme précédent.
- Remplacer UPDATE par MERGE. Remarquer que sur cet exemple simple les deux commandes donnent le même résultat.

- Pour le Update, inverser le '1' et le '2' du tabb pour la variable X, et vérifier les erreurs dans le journal. Rappel : pour trier un tableau, la syntaxe est :

```
Proc sort data=tab; by x;
```

6 Update 2

Sur l'exemple précédent, modifiez le tabb comme suit :

```
data tabb;
input x z t;
cards;
1 5 1
1 3 1
```

Remarquez que la clé "1" est présente deux fois dans ce tableau.

- Faites la mise à jour avec UPDATE
- Faites la fusion avec MERGE et comparez.

Comme vous avez pu le constater, MERGE réalise un produit cartésien sur les observations possédant la même clé, alors que UPDATE réalise des mises à jour successives par les observations correspondantes.

7 Update 3 : les valeurs manquantes

Par défaut, UPDATE ne remplace pas les anciennes données par des données manquantes, au contraire de MERGE.

Modifiez le tableau tabb comme suit :

```
data tabb;
input x z t;
cards;
1 5 1
2 3 .
```

- Réaliser le UPDATE de taba par tabb
- Réaliser le MERGE de taba par tabb et comparer

Pour changer le comportement par défaut de UPDATE, il faut utiliser l'option UPDATEMODE=NOMISSINGCHECK :

```
update taba tabb updatemode=nomissingcheck;
```

8 Update 4 : les valeurs manquantes spéciales

Même lorsque les données manquantes usuelles ne remplacent pas les données du premier tableau, les données manquantes définies par l'utilisateur remplacent les anciennes données. Les valeurs manquantes de A à Z les remplacent telles quelles, la valeur manquante '_' devient une donnée manquante usuelle. Exemple :

```
data taba;
input x y t;
cards ;
1 3 8
2 4 6
;
```

```
data tabb;
missing _ a;
input x y t;
cards;
2 a .
1 3 _
;
data tab;
update taba tabb;
by x;
```

- Taper le programme précédent, en remarquant bien que j'ai pris X Y T comme variables de tabb, et non X Z T. Vous devez aller voir le journal et corriger les erreurs ou manques éventuels.

8 EXPORTER IMPORTER

1 Instruction FILENAME

Nous pouvons définir une référence à un fichier en utilisant l'instruction FILENAME :

```
filename ventes 'C:\temp\ventes.txt';
data tab;
      infile ventes;
      input nb @@;
   run;
```

Pour exécuter ce code, il faut créer un fichier `'C:\temp\ventes.txt'` avec par exemple deux lignes de nombres.
Filename permet de manipuler plus facilement les fichiers, par exemple en simplifiant les références multiples à un même fichier, ou encore en changeant facilement le fichier source.

2 Proc Export

Proc Export permet d'exporter un tableau SAS dans un fichier texte, un fichier excel ou autre. Par exemple pour exporter un tableau en délimitant les données avec le caractère '&', nous écrivons :

```
proc export data=sashelp.class
   outfile='c:\temp\class.txt'
   dbms=dlm replace;
     delimiter='&';
run;
```

- Entrer le programme précédent en remarquant l'absence de point-virgule sur les deux premières lignes.
- Utiliser l'instruction filename pour utiliser une référence de fichier avec la commande OUTFILE :

```
Filename class 'C:\temp\class.txt' ;
proc export data=sashelp.class outfile=class ...
```

3 Dbms

L'option dbms indique quel type de sortie nous souhaitons. Entre autres, nous pouvons spécifier EXCEL, DLM.

- Essayer avec Excel, puis ouvrez le fichier dans un tableur :

```
proc export data=sashelp.class outfile='c:\temp\class.xls' dbms=excel replace;
```

```
run;
```

Remarquez l'extension du fichier ainsi que l'absence de la spécification du délimiteur.

4 Replace

Par défaut, Proc Export ne peut écrire sur un fichier existant. Pour autoriser SAS à remplacer un fichier, il faut utiliser l'option REPLACE. Essayez le programme suivant, puis enlevez l'option REPLACE et allez voir l'erreur dans le journal.

```
proc export data=sashelp.class
   outfile='c:\temp\class.txt'
   dbms=dlm replace;
     delimiter=':';
run;
```

5 Proc Import

Proc Import permet d'importer des données, contenues dans un fichier texte, Excel, ou encore une table de données Access.

Par exemple soit le fichier `"C:\temp\delimiter.txt"` contenant les lignes suivantes :

```
Region&State&Month&Expenses&Revenue
Southern&GA&JAN2001&2000&8000
Southern&GA&FEB2001&1200&6000
```

Nous pouvons le récupérer avec les instructions suivantes :

```
proc import datafile="C:\temp\delimiter.txt"
      out=mydata  dbms=dlm   replace;
      delimiter='&';
      getnames=yes;
run;
proc print data=mydata;
run;
```

La commande OUT indique le nom du tableau SAS créé, REPLACE indique que nous remplaçons le tableau si celui-ci existait déjà.

- Exécuter les instructions précédentes.
- Utiliser FILENAME pour ne pas avoir le nom du fichier dans le *Proc Import,* comme ce que l'on avait fait avec le *proc export* au point 2.

6 Récupérer les noms des variables

Avec Proc Import, il est possible de récupérer les noms des variables d'un fichier texte ou d'un fichier Excel en spécifiant l'option `GETNAMES=YES`.

Si l'on spécifie GETNAMES=NO, ou bien si les entêtes ne sont pas des noms SAS valides, Proc Import crée automatiquement les variables Var1, Var2, ...

7 Sélectionner une feuille dans un classeur

On peut importer une feuille d'un classeur Excel à l'aide de la commande SHEET :

```
proc import datafile="C:/temp/liste.xls"  out=liste;
```

```
sheet='Feuil1';
getnames=no;
run;
```

Il faudra créer un fichier excel avec des données sur la première feuille, en vérifiant si le nom de cette première feuille est bien *Feuil1*.

8 Proc COPY : sauvegarder

```
Libname disk 'C:\temp';
Proc Copy in=work out=disk;
select tab;
quit;
```

9 Proc Copy : récupérer

Il suffit d'adapter l'exemple précédent en intervertissant work et disk.

10 Exporter/Importer un tableau en XML

```
libname myxml xml 'c:\temp\test.xml';
data myxml.hh;
set tab;
run;
```

La première ligne affecte le fichier c:\temp\test.xml' à la bibliothèque myxml. Nous spécifions que nous utilisons le moteur XML pour cette bibliothèque. La bibliothèque 'myxml' est ensuite utilisée comme d'habitude, ici pour exporter en xml.

Les bibliothèques de type XML peuvent être utilisées pour créer des tableaux, les lire. Par contre on ne peut trier ces tableaux ni réaliser de jointure.

9 MANIPULER LES TABLEAUX SAS

1 Proc DATASETS

La procédure DATASETS est un utilitaire vous permettant de gérer vos fichiers SAS. Avec Proc DATASETS, vous pouvez par exemple :

- copier des tableaux SAS d'une bibliothèque à une autre
- renommer des tableaux SAS
- supprimer des tableaux
- lister les tableaux SAS présents dans une bibliothèque

2 Copier des tableaux

Pour copier les tableaux d'une bibliothèque dans une autre, nous utilisons l'instruction COPY :

```
proc datasets library=Sasuser;
copy out=work;
run;
```

- Entrer le programme précédent

3 Supprimer un tableau

```
proc datasets library=work;
```

```
delete Body;
run;
```

4 Modifier les propriétés d'un tableau

Avec DATASETS, nous pouvons modifier les caractérisitiques d'un talbeau, comme les mots de passe, les noms des variables, changer ou spécifier des labels ...

5 Ajouter un mot de passe à un tableau

```
proc datasets;
modify tab (label='test' read=t);
run;
```

A chaque accès du tableau en lecture, SAS demande d'entrer un mot de passe, ici 't'. Le libellé 'test' associé au tableau est visible à l'aide de Proc CONTENTS.

6 Modifier un mot de passe

```
proc datasets;
modify tab (read=t/m);
run;
```

7 Supprimer un mot de passe

```
proc datasets;
modify tab (read=m/);
run;
```

8 Différents niveaux de mot de passe

Il existe quatre niveaux de mot de passe. Je reprend ici l'aide de SAS :

Modify an alter password	ALTER=
Modify a read, write, or alter password	PW=
Modify a read password	READ=
Modify a write password	WRITE=

Pour créer, modifier ou supprimer un mot de passe, il faut au préalable rappeler le mot de passe de plus haut niveau existant pour le tableau.

- Ajouter un mot de passe de type PW au tableau TAB
- Afficher le tableau
- Ajouter un mot de passe de type WRITE au tableau TAB
- Afficher le tableau

9 Read, Write et Alter

Voici une liste d'exemples associés à chaque type de mot de passe :

Read	Proc print data=tab ; Proc means data=tab ; data tab2 ; set Tab ;

Write	Proc sort … Proc append base=tab ...
Alter	Data tab ; x=2 ; run ; Proc datasets ; modify tab ; label x='varx' ;

Le mot de passe Read empêche de lire les données. Pour comprendre les mots de passe Write et Alter, il faut rapeler qu'un tableau SAS est constitué de deux parties : le descripteur qui contient les variables avec leur type, leur longueur et autres labels, et les données. Le mot de passe Write empêche de toucher aux données (par exemple Proc Sort), le mot de passe Alter empêche de toucher au descripteur, comme par exemple ajouter un label à une variable.

10 SQL

Acronyme de Structured Query Language, le langage SQL permet d'accéder rapidement à une base de donnée. Ce langage est utilisé dans les programmes de base de données tels que Access, Oracle ,Mysql, ou encore Postgresql.

1 Création du tableau d 'étude

```
DATA tab;
DO i=1 TO 40 BY 1;
nom='A'; an=1930+i; OUTPUT;
END;
DO j=1 TO 40 BY 1;
nom='B'; an=1950+j; OUTPUT;
END; RUN;
```

2 Lecture d'une base de donnée avec SELECT et FROM

La lecture du tableau du tableau *TAB* précédent s'effectue ainsi :

```
PROC SQL;
SELECT an, nom
FROM tab;
```

Remarquer la virgule séparant les variables *an* et *nom.* Remarquer également l'absence de point-virgule sur la ligne de SELECT : les instructions SELECT et FROM font parties de la même requête.

3 La procédure SQL

En premier lieu on appelle cette procédure par la commande

```
PROC SQL;
```

Les instructions qui suivent sont des instructions SQL. Elles sont organisées en blocs appelés requêtes. Une requête se termine par un point-virgule. La virgule est maintenant le séparateur pour les listes de noms, comme par exemple pour les listes de variables.
Pour sortir de la procédure SQL, nous entrons soit « QUIT; » soit une instruction SAS.

4 Sélection des observations avec WHERE

Nous souhaitons voir uniquement les observations dont le nom est 'A' :

```
Proc sql;
select an,nom from tab WHERE nom='A';
```

5 Session SQL

Une session SQL se termine par un *quit* ou une étape SAS.
Tant que SAS ne rencontre pas de *quit* ou d'étape SAS, vous restez dans la session SQL. En particulier vous devriez voir en haut de votre éditeur *Proc SQL en cours d'exécution.* Il est donc inutile d'écrire plusieurs fois *Proc SQL* : il faut écrire un seul *Proc SQL*, puis la liste des requêtes SQL. De plus, tant que vous êtes dans la session SQL, il est inutile de relancer le *Proc SQL* : vous pouvez lancer individuellement chacune de vos requêtes SQL.

Notez que le *run* est inutile : une requête SQL est lancée immédiatement après avoir été lue par SAS. La programmation SQL est une programmation par requête, à la différence de SAS qui est une programmation par étape.

Avec SQL, nous pouvons faire toutes les manipulations de données que nous faisions avec des étapes DATA. Cependant ce cours reste un cours de SAS : en examen, sauf mention contraire, toute question doit être résolue avec le langage SAS, et non en SQL.

6 Tri d'une base de donnée avec ORDER BY

Ajouter l'instruction ORDER BY à la fin de l'instruction SELECT pour ordonner les observations selon les modalités d'une variable :

```
Select an, nom from tab order by an;
```

7 Ajout de variables

On entre les nouvelles variables calculées à partir de celles existantes dans l'instruction SELECT :

```
SELECT an, nom, an-1900 AS annee from Tab ;
```

L'instruction AS spécifie le nom de la nouvelle variable.

8 Création d'un tableau permanent

Pour créer un tableau permanent, nous entrons juste après PROC SQL :

```
CREATE TABLE tabl AS
```

Par exemple pour créer le tableau correspondant à notre premier programme SQL, nous entrons

```
Proc sql;
CREATE TABLE tabl AS
SELECT an, nom
FROM tab;
```

9 Fusion de tableaux

Un gérant de stock considère les données suivantes :

Id_prod	Nom_prod
1	bol
2	verre
3	couteau
4	fourchette
5	assiette

Id_prod	Quantite
2	4
3	2
4	2

Créez à l'aide de deux étapes Data ces tableaux. Pour le tableau *produits*, le input sera :

```
input id_prod nom_prod $:12.;
```

Pour fusionner les deux tableaux *produits* et *quantites*, nous écrivons ensuite :

```
Proc sql;
select produits.id_prod, nom_prod, quantite
from produits, quantites
WHERE produits.id_prod=quantites.id_prod;
```

3.Le résultat ne contient que les observations dont *Id_prod* est présent dans les deux tableaux. Une telle fusion s'appelle **jointure interne**.
4.Nous fusionnons les observations dont les valeurs de *Id_prod* sont égales dans les deux tableaux.
5.Dans l'instruction SELECT, nous utilisons la syntaxe *nom_tableau.nom_variable* pour faire le distinguo lorsque ce nom de variable est utilisé dans les deux tableaux. Nous pouvons utiliser cette syntaxe pour toutes les variables, cela permettant de rendre plus lisible le programme.

10 Mettre des alias

Il est parfois fastidieux de rappeler à chaque fois le nom complet des tableaux. On peux utiliser des alias comme suit :

```
Proc sql;
select P.id_prod, nom_prod, quantite
from produits P, quantites Q
where P.id_prod=Q.id_prod;
```

Nous avons créé les alias lors de l'instruction FROM, et nous les avons utilisés dans les instructions SELECT et WHERE. Ici « P » remplace « produits » et « Q » remplace « quantites ». Le résultat est identique, seul le code a changé.

11 Combiner trois tableaux

Il est tout à fait possible de faire une fusion interne sur trois tableaux :

```
Data TabPoids ; input id poids @@; cards ;
1 120 2 70 3 40
;
proc sql ;
select p.id_prod, nom_prod, quantite, poids from produits P, quantites Q, tabPoids T where
P.id_prod=Q.id_prod and P.id_prod=T.id ;
```

12 Jointure externe

Dans certains cas nous souhaitons garder toutes les observations d'un tableau. Par exemple notre gérant de stock aime connaître la quantité de ses ustensiles, mais aimerai avoir en même tous les

ustensiles gérés par son magasin. Pour ce faire, nous entrons

```
Proc sql;
select P.id_prod, nom_prod, quantite
from produits P LEFT JOIN quantites Q
ON P.id_prod=Q.id_prod;
```

- Les mots clefs sont LEFT JOIN et ON.
- L'instruction LEFT JOIN précise que nous gardons toutes les observations du tableau de gauche, ici le tableau *produits*. Remarquer que cette instruction remplace la virgule de la jointure interne.
- L'instruction ON vient remplacer l'instruction WHERE lors d'une jointure externe
- A la place de LEFT JOIN, les autres possibilités sont RIGHT JOIN et FULL JOIN.
- Une jointure externe ne s'effectue que sur deux tableaux.

11 STATISTIQUES DESCRIPTIVES

1 PROC MEANS

La procédure MEANS permet d'obtenir les statistiques les plus courantes. Appliquons cette procédure à un tableau aléatoire :

```
data etude;
do x=1 to 20;
y=rand('binomiale',0.4,20);
output;end;
proc means;
run;
```

- Entrer les instructions précédentes.

2 Les options de Proc Means

Nous présentons les mots clefs correspondants aux différentes statistiques disponibles. Commençons par les statistiques données par défaut : la taille de l'échantillon (**n**), la moyenne arithmétique (**mean**), l'écart-type (**std**), la valeur minimale (**min**) et la valeur maximale (**max**).

Les autres statistiques sont : le nombre de données manquantes (**nmiss**), l'étendue (**range**), la somme (**sum**), la somme des carrés (**uss**), la somme des carrés corrigée (**css**), la variance (**var**), l'erreur-type de la moyenne (**stderr**), le coefficient de variation (**cv**), le coefficient d'asymétrie (**skewness**) et le coefficient d'aplatissement (**kurtosis**).
Exemple :

```
Proc Means n sum skewness ;
run ;
```

3 Commande FREQ

La commande Freq dans Proc MEANS permet de spécifier l'effectif de la modalité correspondante. Par exemple pour étudier la distribution suivante

Nb enfants	0	1	2	3	4
Effectif	8	18	17	9	3

nous entrons :

```
Data tab ; input nb eff @@; cards ;
```

```
0 8 1 18 2 17 3 9 4 3
;
proc means ; var nb ; freq eff ; run ;
```

Trouver maintenant avec *proc means* et *freq* la moyenne de la taille pour la distribution suivante :

Taille	[140,160[	[160,170[	[170,186[	[186,200[
Effectif	8	17	14	11

Indication : il faudra entrer dans deux variables distinctes les bornes des intervalles et calculer les centres des classes ; vous aurez ainsi les deux lignes suivantes dans votre étape DATA :

```
Input t1 t2 n ;
c=(t1+t2)/2 ;
```

4 Les commandes BY et CLASS

Pour avoir des statistiques sur différents groupes de la population, nous pouvons utiliser BY ou CLASS. Le tableau *Tab5* suivant nous servira également par la suite.

```
Data tab5 ; do x=1, 2 ;
do i=1 to 10 ; y=x+rannor(3) ; z=x+1+rannor(5) ; output ; end ; end ;
proc means ; var Y ; by X ;
proc means ; var Y ; class X ; run ;
```

La fonction *rannor* permet d'obtenir le tirage d'une variable normale (gaussienne centrée réduite). Les deux sorties sont assez proche. La commande BY impose que les données soient préalablement triées, ce qui n'est pas le cas avec la commande CLASS.

5 Sélection des variables avec VAR

Comme avec Proc Print, on peut sélectionner une liste de variable en les indiquant dans la commande *var* :

```
proc means ; var Y Z ; class X; run;
```

6 La commande TYPES

Lorsque l'on spécifie plusieurs variables avec CLASS, SAS réalise un produit cartésien des variables pour obtenir un découpage. Il est possible de spécifier plusieurs découpages avec TYPES. Commençons par créer un tableau d'étude :

```
Data tab;
input pays $ @@;
do annee=1950 to 1960;
do genre='F', 'M';
taille=166+7*rannor(5);/*ne vous occupez pas du 5 dans rannor*/
output; end; end;
cards;
France Brésil Allemagne
;
```

Entrez maintenant le programme suivant :

```
Proc Means;
var taille;
class annee pays genre;
```

```
types annee pays*genre ();
run;
```

Nous obtenons les informations selon plusieurs découpages : par année, par pays et genre, et pour la population totale avec les parenthèses. Ces instructions donnent donc trois tableaux. L'un a une seule ligne, et correspond à la population totale comme demandé avec les parenthèses vides, le deuxième a six lignes pour pays*genre, car la variable pays a trois modalités et le genre en a deux, et le dernier tableaux a onze lignes, une pour chaque année.

7 PROC UNIVARIATE

Cette procédure permet d'obtenir un nombre plus important de statistiques. Nous pouvons obtenir les valeurs extrêmes, la médiane et les centiles, la distribution des fréquences et des diagrammes.

- **L'option PLOT**

Avec l'option plot, nous obtenons un diagramme en feuilles, un diagramme en boîte et une courbe de probabilité normale cumulée. Il faut écrire

```
Proc univariate plot data=tab5 ;/*le tab5 du point 4*/
var Y ; run ;
```

- **L'option FREQ**

Avec l'option freq, nous obtenons la liste des valeurs de la variable, la fréquence de ces valeurs, le pourcentage, le pourcentage de ces valeurs et le pourcentage cumulé de ces valeurs.
Les options Plot et Freq se place sur la même ligne que Proc Univariate, par exemple

```
Proc univariate freq data=tab5 ; var Y ; run ;
```

8 PROC CORR

La procédure corr permet d'obtenir les coefficients de corrélation linéaire. Nous obtenons d'abord les statistiques de base pour chaque variable, puis les coefficients de corrélation linéaire.

```
Proc corr data=tab5 ; var Y Z ; run ;
```

9 PROC FREQ

Cette procédure permet d'obtenir une distribution des fréquences.

```
Data tab ; do i=1 to 40 ;
x=ranpoi(6, 2) ;/*variable de poisson de paramètre 2*/
output ; end ;
proc freq;
tables x;
run;
```

Nous spécifions, à l'aide de l'instruction tables, les variables pour lesquelles nous désirons une distribution des fréquences. L'instruction 'x*y' permet d'avoir une distribution des fréquences de la loi jointe.
Pour le programme suivant, nous utilisons le tableau suivant :

X	Y	C
1	2	3
1	1	5
2	2	2
2	1	6

Ce tableau représente une distribution des effectifs pour le couple (X,Y). Nous utilisons ensuite

l'instruction WEIGHT pour indiquer que la variable C représente les effectifs :

```
Proc freq;
tables x*y;
weight c; run;
```

Il y a quatre lignes de valeurs dans chaque cellule du tableau obtenu. La première ligne indique l'effectif, la deuxième la proportion de couple de modalités dans la population totale, la troisième ligne est la distribution conditionnelle de Y par rapport à X, et la quatrième ligne la distribution conditionnelle de X par rapport à Y. Il y a par exemple 3 personnes qui vérifient X=1 et Y=2, ce qui représente 18,75 % (=3/16) de la population totale.
Par défaut les valeurs sont ordonnées par ordre croissant. Si nous désirons que l'affichage suive l'ordre d'apparition des données, nous spécifions l'option **order=data**. Si nous spécifions **order=freq**, les valeurs sont présentées suivant leurs fréquences. Par exemple

```
Proc freq order=data; ...
```

12 ANALYSE DE DONNÉES : RÉGRESSION

1 Modele linéaire simple

Considérons un modèle de régression simple qui lie deux variables Y et X. On cherche à expliquer linéairement l'évolution de Y en fonction de X.

```
/*données d'entrée pour la
régression*/
DATA tabxy ;
INPUT vx vy ;
Cards;
35 114
45 124
55 143
65 158
75 166
;
run;
```

```
/*visualisation*/
PROC gplot data=tabxy ;
Plot vy*vx;
Run;

/* régression linéaire simple*/
PROC REG data=tabxy;
MODEL vy=vx;
run;

/* régression linéaire simple avec
affichage*/
PROC REG data=tabxy;
MODEL vy=vx;
plot vy*vx='*';
run;
```

2.Taper les instructions suivantes, vérifier par affichage du tableau que les données ont été correctement rentrées dans le tableau puis effectuer un régression linéaire simple de Y sur X.
3.L'instruction « model y=x » définit y comme variable à expliquer et x comme variable explicative.

Etudier les sorties proposées par SAS. Qu'en concluez-vous ?

4.L'instruction « plot » permet d'afficher la droite de régression ainsi que les points de l'échantillon. Le graphique montre clairement une dépendance linéaire entre les variables, confirmé par le coefficient de corrélation.

2 Sortie commentée

Une partie de la sortie obtenue est :

Résultats estimés des paramètres

Variable	DF	Résultat estimé des paramètres	Erreur std	Valeur du test t	Pr > |t|
Intercept	1	65.10000	5.82838	11.17	0.0015
vx	1	1.38000	0.10263	13.45	0.0009

La droite de régression de vy par rapport à vx est donc :
vy=65,1+1,38*vx.

3 Modification des variables

Parfois il n'y a pas de relation linéaire entre Y et X, mais plutôt entre Y et X^2, ou encore entre ln(Y) et X. Pour étudier cette dépendance linéaire, il faut introduire une nouvelle variable puis prendre le modèle correspondant.

4 Modele linéaire simple (2)

Taper les instructions suivantes, afficher le nuage de points, puis effectuer une régression linéaire simple. Visualiser les résidus en fonction de la variable à expliquer Y. Qu'en concluez-vous ?

Le répertoire 'c:\tmp' pourra être remplacé par une autre adresse indiquée en cours.

```
/*données d'entrée pour la
régression*/
DATA fic2;
do vx=1 to 4 by 0.4;
vy=exp(vx)+0.1*ranuni(5);
output; end;
run;
```

```
/* régression linéaire simple*/
PROC REG data=fic2;
MODEL vy=vx;
run;
```

Effectuer un changement de variable adéquate, puis effectuer à nouveau une régression linéaire sur les variables transformées. Quel(s) indicateur(s) vous permettent de justifier le changement de variable effectué ?

```
/*création de la variable lnvy*/
data fic2;
set fic2;
lnvy=LOG(vy);
run;
```

```
/* régression simple avec affichage*/
PROC REG data=fic2;
MODEL lnvy=vx;
plot lnvy*vx='*';
run;
```

13 PROC TABULATE

1 Présentation

Proc Tabulate est un procédure de Base/SAS qui permet de donner des statistiques sur une variable, en découpant la population suivant plusieurs critères. Sa force principale est d'obtenir un tableau découpant la population de manière intuitive.

2 Le tableau d'étude

Créons en premier lieu un tableau d'étude :

```
data energy;
input region division @@;
do type=1 to 2;do i=1 to 8;
expenditures=500+50*rannor(2);
output; end; end;
cards;
1 1 1 2 4 3 4 4
;
```

3 Un exemple simple

```
proc tabulate;
   class region division type;
   var expenditures;
   table region*division, type*expenditures;
run;
```

- Entrer le programme précédent.

Par défaut, chaque cellule du tableau contient la somme de la variable étudiée, ici expenditures, pour chacune des sous-populations correspondantes.

4 Le type Class et le type Var

Les variables déclarées après la commande CLASS sont les critères qui vont découper notre population.
Les variables déclarées après la commande VAR sont les variables que l'on désire étudier.
Chaque cellule ne peut correspondre qu'à au plus une seule variable de type VAR.

- Retirer la variable division dans la commande CLASS et regarder le message d'erreur.
- Déclarer la variable division dans la commande VAR et regarder le message d'erreur.
- Déclarer la variable expenditures à la fois dans CLASS et dans VAR et regarder le message d'erreur.

5 La commande Table

La commande TABLE indique le découpage effectif que nous souhaitons appliquer.

- Supprimer certaines variables dans cette commande, en particulier les variables de classe et observer les résultats.
- Faites ces différents tests en ajoutant de nouvelles commandes TABLE.
- Essayer en particulier les commandes suivantes :

```
table region*expenditures, division;
table region division, expenditures;
table region division, type;
table region, division*expenditures expenditures type;
```

Lorsqu'une cellule ne comporte pas de variable de type VAR, SAS indique l'effectif de la population correspondant à cette cellule.

6 Les entêtes

Par défaut les noms des variables apparaissent dans les entêtes des lignes et des colonnes. A chaque variable on peut associer la chaîne de caractère que l'on souhaite afficher :

```
table region='Les régions'*division='Les divisions',
type='Le type' *expenditures='Variable étudiée';
```

7 Dimension d'un tableau

Dans la commande TABLE on entre des produits de variables, séparés par des virgules. Chacun de ces produits correspond à une dimension du tableau. Un tableau peut être à une, deux ou trois dimensions.
Si un tableau n'a qu'une dimension, cette dimension est disposée en colonne.
Si un tableau a deux dimensions, la première correspond aux lignes, la seconde aux colonnes.
Si un tableau a trois dimensions, la première est divisée en page, la seconde correspond aux lignes et la troisième aux colonnes.

- Interchanger le premier produit et le second dans la commande TABLE.
- Essayer

```
table region*expenditures;
```

- Essayer

```
table region, division, type*expenditures;
```

8 Un exemple plus compliqué

```
proc tabulate data=energy;
    class region division type;
    var expenditures;
    table region*(division all='Subtotal')
         all='Total for All Regions',
         type*expenditures
         all='All Customers'*expenditures;
run;
```

- Entrer le programme précédent.
- Après la virgule du TABLE, spécifiez :

```
type*expenditures type expenditures
         all='All Customers'*expenditures
```

Regardez la différence entre la colonne Expenditures et la colonne all*Expenditures.

9 Mot clé ALL

En entrant une variable dans la commande Table, on indique un découpage. Le mot clé ALL est comme une variable de type Class mais qui ne découpe pas la population.
Dans l'exemple précédent, on divise les Régions par la variable Division, mais également par la variable ALL, ce qui permet d'avoir par Région les informations par Divisions et également les informations pour la région entière dans la ligne 'SubTotal'.
Avec le mot clé ALL on peut créer un tableau de dimension 1 et disposé en ligne :

```
table type*expenditures, all;
```

10 Simplifier l'affichage

Nous savons que nous étudions la variable Expenditures et nous ne voulons pas la voir apparaître dans le tableau :

```
table region*(division all='Subtotal')
         all='Total for All Regions',
         type='Customer Base'*expenditures=' '
         all='All Customers'*expenditures=' ';
```

Sur les colonnes nous voyons à chaque fois apparaître le mot « Sum », et dans le cas présent ceci est

superflu. Pour l'enlever nous utilisons

```
table region*(division all='Subtotal')
         all='Total for All Regions',
         type='Customer Base'*expenditures=' '*sum=' '
         all='All Customers'*expenditures=' '*sum=' ';
```

Ainsi Proc Tabulate n'affiche pas les entêtes ne contenant qu'un espace.

11 Appliquer un format

Pour appliquer un format on utilise :

```
*f=dollar12.
```

dans une définition de ligne ou de colonne.
Ainsi un exemple complet est :

```
table region*(division all='Subtotal')
     all='Total for All Regions'*f=dollar12.,
     type='Customer Base'*f=dollar12.*expenditures=' '*sum=' '
     all='All Customers'*expenditures=' '*sum=' ';
```

Lorsque l'on essaie d'appliquer plusieurs formats à un cellule, le comportement par défaut est de prendre le dernier format spécifié.

- Entrez l'exemple, puis lors de la déclaration d'une colonne spécifiez le format 12.2.

12 Les statistiques disponibles

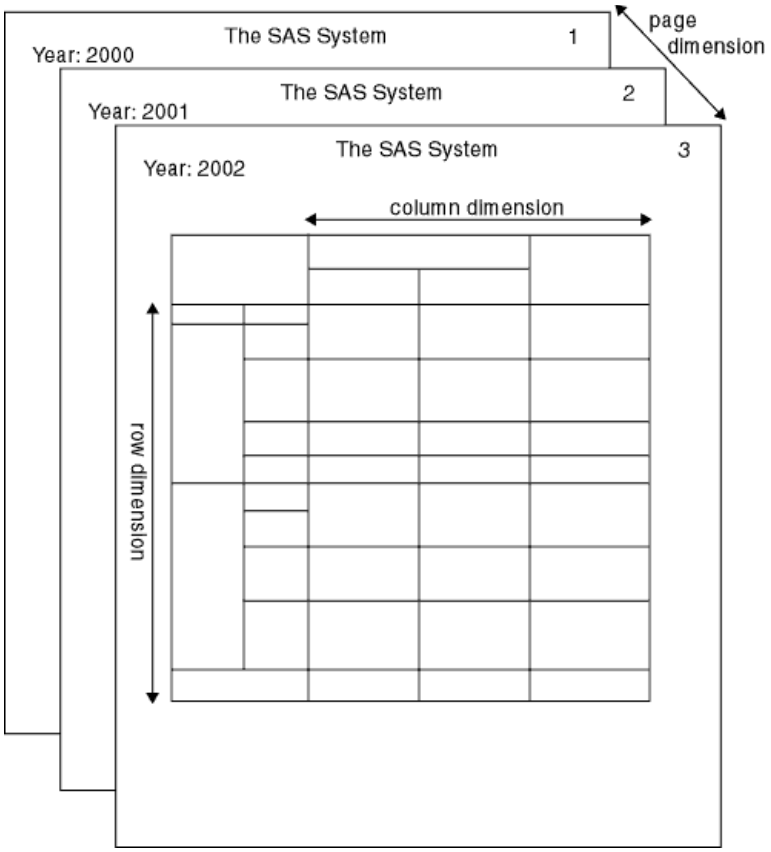
Par défaut Proc Tabulate affiche dans chaque cellule la somme de la variable étudiée, mais il en existe bien d'autres.
Pour la liste des statistiques complète disponibles, voir
Sommaire → SAS Products → Base SAS → SAS Procedures → The Tabulate Procedure → Concepts: TABULATE Procedure
Nous retiendrons en particulier N, SUM, MAX, MIN, COLPCTN, COLPCTSUM, ROWPCTN, ROWPCTSUM.

13 Spécifier des statistiques

Nous voulons par exemple pour chaque cellule la somme (nous l'avions déjà), le nombre d'observations, la valeur maximale, le pourcentage que représente cette cellule dans la ligne en terme de nombre d'observations, puis le pourcentage que représente sa somme par rapport à la somme de la ligne. Nous entrons :

```
proc tabulate data=energy;
    class region division type;
    var expenditures;
    table region*(division all='Subtotal')
      all='Total for All Regions',
      type*expenditures*(sum N='nombre' max ROWPCTN ROWPCTSUM)
         all='All Customers'*expenditures;
run;
```

Représentation des dimensions



14 OUTPUT DELIVERY SYSTEM

Par défaut la sortie de SAS est un listing apparaissant sur la fenêtre Sortie. Le module ODS de SAS permet d'obtenir d'autres types de sortie, comme un fichier HTML, un fichier RTF ou encore PDF.

Destinations SAS formatées

Pour obtenir un nouveau format, on ouvre la sortie avec par exemple « ods html », on exécute la procédure voulue, puis à la fin on referme cette sortie afin de ne pas surchargé SAS.

1 Une sortie PDF :

```
data tab;
      input Group $ Score @@;
      datalines;
   A 75  A 76  A 80  A 77  A 80  A 77  A 73
   B 82  B 80  B 85  B 85  B 78  B 87  B 82
   ;

   ods pdf file='ttest.pdf';
   title 'Comparaison de la moyenne des groupes';
   proc ttest ;
      class Group;
      var Score;
   run;
   ods html close;
```

2 Créer un fichier RTF :

```
ODS RTF FILE='ttest.rtf';
proc univariate data=tab;
run;
ODS RTF CLOSE;
```

3 Intégrer l'ODS dans une étape DATA :

```
ODS RTF FILE='data_step.rtf';
DATA Tab2;
SET Tab;
FILE PRINT ODS;
PUT _ODS_;
RUN;
ODS RTF CLOSE;
```

Sélection des tables affichées

4 Obtenir les noms des tables ODS

```
ods trace on / listing;
        proc univariate data=Tab;
        run;
ods trace off;
```

On spécifie ainsi que la trace de chaque table doit être affichée. L'option listing indique que cette information doit s'intercaler avec le listing classique.

5 Sélectionner les tables pour l'affichage

Proc univariate a par exemple la sortie Quantiles. Vous pouvez donc exécuter :

```
ods select quantiles;
ods trace on;
ods show;
proc univariate data=tab;
run;

ods select all;
```

Ainsi on sélectionne une table. Avec ODS SHOW, le journal contient les tables sélectionnées, et avec ODS TRACE ON le journal contient les tables crées lors de l'exécution.

6 Exclure des tables pour l'affichage :

```
ods exclude Means(persist);
```

L'option persist indique que cette table sera exclue pour toute la session SAS, à moins que l'on n'utilise un ods select ou bien ods exclude none.

7 Créer un tableau SAS à partir d'une sortie :

```
Data tab;
do score=1 to 10;output; end;
run;
ods output summary=mytab;
```

```
proc means; run;
proc print data=mytab;
run;
ods output summary=mytab2 (keep =Score_Mean);
proc means data=tab; run;
proc print data=mytab2;
run;
```

Cela écrit la table ODS dans un tableau SAS appelé mytab. On peut lors de cette création spécifier les variables à garder et également en renommer. On peut également placer une clause WHERE pour sélectionner des observations.

8 ODS Document

La sorite "Document" est une sortie spécifique qui contient toutes les informations de sorties de la procédure.

```
data tab;
do i=1 to 40;
x=3+rannor(2);
y=ranbin(6, 2, 0.4);
output; end;
run;
ods document name=testdoc;
proc print; run;
proc means;
var x;
run;
proc means;
var x;
class y;
run;
ods document close;
```

On peut parcourir ce document en entrant dans la ligne de commande : ODSDOCUMENTS.

- Entrer cette commande et parcourir le document
- Effacer la sortie et observer que les résultats contenus dans le document sont accessibles et se rajoutent à la fenêtre des résultats.

9 Proc Document

La procédure Proc Document permet de lister le contenu d'un document, placer des titres et des

notes, contrôler les sauts de pages, envoyer les résultats dans diverses sorties. Une fois déclarée, la procédure se termine par un QUIT, l'instruction RUN ne permettant pas de stopper la procédure.

10 Lister le contenu d'un document

Pour lister le contenu d'un document, nous utilisons l'instruction LIST :

```
proc document;
   doc name=testdoc;
   list/levels=all;
run;
quit;
```

La deuxième ligne permet de spécifier à l'intérieur de la procédure le document utilisé. Nous aurions également pu écrire :

```
proc document name=testdoc;
   list/levels=all;
run;
quit;
```

11 Créer un fichier PDF à partir d'une sélection des résultats

La commande REPLAY permet d'envoyer dans les sorties actives un ou plusieurs résultats :

```
proc document name=testdoc;
ods pdf file='monfichier.pdf';
replay means#2\summary#1;
replay print#1\print#1;
replay means#1\summary#1;
run;
quit;
ods pdf close;
```

Sans les sauts de page intempestifs :

```
proc document name=testdoc;
ods pdf file='monfichier.pdf' startpage=no;
replay means#2\summary#1;
replay print#1\print#1, means#1\summary#1;
run;
quit;
ods pdf close;
```

12 Titre, sous-titre, notes, pieds de page

Nous pouvons utiliser comme précédemment le titre par TITLE et le note de pied de page par FOOTNOTE.

Voici la liste des commandes supplémentaires :

OBANOTE	
OBBNOTE	
OBFOOTN	
OBPAGE	
OBSTITLE	
OBTITLE	Identique à TITLE

Entrez :

```
proc document name=testdoc;
ods pdf file='monfichier.pdf' startpage=no;
obanote print#1\print#1 'obanote pour print';
obbnote print#1\print#1 'obbnote pour print45';
replay means#2\summary#1;
replay print#1\print#1, means#1\summary#1;
run;
quit;
ods pdf close;
```

Puis complétez en :

```
proc document name=testdoc;
ods pdf file='monfichier.pdf' startpage=no;
obanote print#1\print#1 'obanote pour print';
obbnote print#1\print#1 'obbnote pour print45';
obfootn means#2\summary#1 'obfootn pour means2';
obfootn1 print#1\print#1 'obfootn pour print';
obfootn3 print#1\print#1 'obfootn3 pour print';
replay means#2\summary#1;
replay print#1\print#1;
replay means#1\summary#1;
run;
quit;
ods pdf close;
```

13 Importer dans un document

Nous pouvons importer un tableau dans un répertoire d'un document :

```
proc document name=class;
import data=tab to \Contents#1\DataSet#1;
run;
```

```
quit;
```

Dans cet exemple, nous importons le tableau Work.tab dans le répertoire spécifier après le mot clé TO.
Le tableau est importer à la fin du répertoire. Pour le mettre au début, nous utilisons :

```
proc document name=class;
import data=tab to \Contents#1\DataSet#1 /first;
run;
quit;
```

Par défaut, l'option est placée à LAST.
Pour un placement plus fin, nous utilisons before= ou after= suivi du nom d'un chemin (qui est à l'intérieur du répertoire considéré).

15 LANGAGE MATRICIEL SOUS SAS

1 Présentation

SAS IML est un modèle spécialisé interprétant un langage de calcul matriciel. L'objet de base de manipulation du langage est une matrice, un tableau bidimensionnel (nrow X ncolumn) de valeurs numériques ou de caractères. Le langage IML permet ainsi de programmer des procédures non disponibles dans SAS.
Une table SAS peut être sauvegardée dans une matrice, ou inversement créée à partir d'une matrice.
SAS IML commence par l'instruction **PROC IML ;** et se termine par **QUIT ;**.

2 Un petit programme pour commencer

```
Proc IML;
X={2 3, 1 4};
Y=2*X;
print X Y;
```

3 Session IML

Une fois que l'on quitte la session IML, que ce soit avec un *quit* ou une étape SAS, toutes les matrices disparaissent. La section suivante vous indique en particulier comment sauvegarder une matrice dans un tableau SAS, mais le principe à retenir est de ne pas quitter la session IML et placer toutes les opérations matricielles dans un seul Proc IML.

4 Interaction entre SAS IML et Base SAS

Transfert d'une table SAS vers une matrice IML.

Pour copier les variables v1 v2… vn dans la matrice X ;
PROC IML ;
Use nomtable ;
READ ALL VAR {v1 v2 … vn} INTO X ;
QUIT;

Transfert de données d'une matrice IML vers une table SAS

PROC IML;
CREATE nomtable FROM X ;
APPEND FROM X;
CLOSE nomtable;

QUIT;

Exemple complet

```
Data tab1;
input x y @@; cards;
8 7 1 3
;
PROC IML ;
        Use tab1 ;
READ ALL VAR {x y} INTO X ;
PRINT X;
CREATE tab10 FROM X ;
APPEND FROM X;
CLOSE tab10;
QUIT;
Proc Print data=tab10;
run;
```

Vérifiez en particulier que les variables de Tab10 sont nommées COL1, COL2 ...

5 Quelques instructions du langage matriciel IML:

PRINT : affiche le contenu de la matrice X.

X={ 1 2, 3 4 } définit la matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

Y=X` ou Y=T(X) transpose la matrice X.
X*Y : produit matriciel.
X#Y : produit élément par élément.
X||Y : juxtaposition de deux vecteurs colonnes.
X[i,j] : accès à la valeur ixj.
X[,+] : vecteur colonne contenant la somme de chaque ligne, X[,:] contient la moyenne de chaque ligne.
X[:,] : vecteur ligne contenant la moyenne de chaque colonne.
X[#,] : vecteur ligne contenant le produit de chaque colonne.
X[##,] : vecteur ligne contenant la somme des carrés pour chaque colonne.
J(n1,n2,val): crée une matrice de n1 lignes et de n2 colonnes dont tous les éléments prennent la valeur val.
I=I(n) : crée la matrice identité de taille nxn.
DIAG : crée une matrice diagonale dont les éléments sont les termes diagonaux de la matrice argument.
VECDIAG : crée un vecteur dont les éléments sont les éléments diagonaux de la matrice argument.
EIGVEC : retourne la matrice des vecteurs propres.
EIGVAL : retourne la matrice des valeurs propres.
TRACE : retourne la trace de la matrice.
DET : calcule le déterminant de la matrice.
INV : calcule l'inverse d'une matrice.
Y=X**alpha : calcul de la puissance alpha de la matrice. Par exemple X**3 est le cube de la matrice X.

ncol(X) : nombre de colonnes de X.
nrow(X) : nombre de lignes de X.
normal(X) crée une matrice de même taille que X contenant des variables normales. La matrice X n'intervient que par sa taille et non par ses valeurs.
do i=1 to N ;…. ;end; : instruction de boucle.
If … then … ; else…; instructions conditionnelles.

6 Un programme complet

```
Proc IML ; X={2 3, 1 4};
Y=2*X;
print X Y;
Z1=eigval(X) ; Z2=eigvec(X) ;
Z3=X||Y ; Z4=normal(X) ;
print Z1 Z2 Z3 Z4;
```

La matrice Z1 indique les valeurs propres de X ; elle a deux colonnes, la première indiquant la partie réelle, la seconde la partie imaginaire des valeurs propres.
Remarquez bien que l'on affiche des matrices, pas des résultats de calculs. Par exemple on ne peut pas écrire *print 3*X.*

7 Exercice d'application : géneration de variables aléatoires gaussiennes

Exécuter et analyser les instructions suivantes ;

```
PROC IML ;
u=repeat(0,100,1) ;
x=normal(u) ;
print U X
quit ;
```

```
PROC IML ;
u=repeat(0,100,1) ;
x=normal(u) ; y=normal(u);
z=x||y;
print z;
```

Ajoutez :

```
create tab11 from z;
append from z;
close tab11;
```

Puis exécutez :

```
proc gplot data=tab11;
plot col2*col1;
run;
```

16 SURVOL DU MACRO LANGAGE SOUS SAS

1 Un exemple

```
%macro boucle(num);
%do i=1 %to &num;
proc print data=tab&i; run;
%end;
%mend;
%boucle(3);
```

Ce programme crée la macro "boucle" qui a un paramètre, puis appelle cette macro pour affiché tab1, tab2, tab3. Ce programme suppose que les trois tableaux Tab1 à Tab3 sont déjà créés.

2 Explications

Pour créer une macro, nous avons commencé par "%macro nomDeLaMacro;" puis terminé par "%mend". Pour utiliser la macro nous entrons "%nomDeLaMacro".

A l'intérieur de la macro, nous utilisons le langage SAS et le macro-langage. Le macro-langage est repéré par les % et les & qui précèdent un nom.

Ici "i" et "num" sont des macro-variables. Pour les initialiser nous les écrivons tels quels, et pour les utiliser nous les précédons du signe "&".

Les instructions %macro, %mend, %do sont des macro-instructions. Elles permettent de gérer le déroulement du macro-programme.

3 Le macro-langage

Le macro-langage est constitué de :

•macro-variables

•macro-instructions

•macro-fonctions

•macros

4 Les macros-variables

Une macro-variable contient une chaîne de caractère. Elle s'utilise avec le signe "&". Nous avons vu dans l'exemple la macro-variable "num" créée car étant un paramètre de la macro, et la macro-variable "i" créée lors de l'instruction %do. Voici deux autres manières pour créer des macros-variables :

1.Avec %let.

```
%let num2=38;
```

2.Avec CALL SYMPUT, lors d'une étape DATA

```
Data tab;
i=4 ;
call symput("pos", i); run ;
%put &pos;
```

17 LA PROCÉDURE PROC GPLOT

1 Descriptif

La procédure GPLOT permet de tracer tous types de graphes. On précise le type du graphe par l'instruction qui commande le tracé du graphe. Cette instruction peut être BUBBLE, PLOT, ...
Les options du graphe sont indiquées après l'instruction du tracé. Elles en sont séparées par le caractère '/'.

2 Graphe de bulles

Commençons par un exemple de graphe avec l'instruction BUBBLE :

```
data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Petro 18444 34833
;
proc gplot data=jobs;
   format dollars dollar9.;
   bubble dollars*eng=num;
run;
```

On place des bulles avec la variable *dollars* fonction de la variable *eng*. La superficie des bulles est proportionnelle à la variable *num.*

- Entrer les instructions précédentes.

3 Titre et sous-titre

- Vous exécuterez le programme après chaque instruction ajoutée.

Pour agrémenter notre graphe, nous ajoutons un titre et un sous-titre :

```
title1 'Profile des Membres';
title2 'Salaires et Nombre des Membres Ingénieurs';
```

Nous augmentons la taille du titre à l'aide de GOPTIONS :

```
goptions  gunit=pct htitle=6;
```

Puis la taille du sous-titre en réglant la hauteur du texte (à ajouter dans goptions) :

```
htext=4
```

La fonte du texte n'est pas satisfaisante et l'on rajoute l'option

```
ftext=swiss
```

4 Notes de bas de page

Nous indiquons le nom du programme dans une note de bas de page :

```
footnote h=3 j=r 'GPLBUBL1 ';
```

L'option 'h=3' donne la taille des caractères de la note. L'option 'j=r' indique que cette note sera placée à droite. Les autres possibilités sont 'j=l' pour placer à gauche et 'j=c' pour placer au centre.

5 Offset pour les axes

Nous constatons que le cadre coupe les bulles situées à gauche et à droite. Nous agrandissons l'axe horizontal en spécifiant avant le Proc Gplot :

```
axis1 offset=(5,5);
```

Puis nous utilisons cet axe lors de l'instruction bubble :

```
bubble dollars*eng=num/ haxis=axis1;
```

6 L'instruction AXIS

On peut donc définir l'aspect des axes en dehors de la procédure gplot par l'instruction AXIS. On peut définir jusqu'à quatre-vingt-dix-neuf axes par axis1,..., axis99. On appelle ensuite l'axe voulu en spécifiant les options haxis et vaxis lors de l'instruction de tracé :

```
bubble dollars*eng=num/ haxis=axis1 vaxis=axis2;
```

Pour utiliser cet exemple il faut qu'axis1 et axis2 ait été défini.

7 Aspect du cadre

L'aspect du cadre est contrôlé par les axes. Ainsi pour avoir un cadre plus épais de couleur bleue :

```
axis1 color=blue width=3;
```

- Entrer une couleur et une épaisseur différentes pour l'axe vertical et constater les règles de priorité.

Nous décidons d'enlever le nom des variables afin d'alléger le graphe :

```
axis1 label=none;
```

!! Lorsque l'on défini axis1, on écrase les définitions précédentes de cet axe. Pour spécifier plusieurs critères, il faut le faire lors de la même définition.

- Enlever les labels pour les deux axes.

8 Echelle des axes et taille des valeurs

Changeons l'échelle verticale pour afficher les dollars de dix-mille en dix-mille :

```
axis2 order=(0 to 40000 by 10000);
```

- Entrer cette commande
- Spécifier un ordre descendant (il faudra utiliser le signe moins).
- Essayer la suite

```
(0 10000 25000 30000 40000 )
```

9 Gestion des marques

Nous augmentons la taille des traits utilisés pour les marques en ajoutant dans l'instruction axis:

```
major=(height=1.5) minor=(height=1)
```

Les marques mineures sont trop nombreuses. Nous réglons leur nombre par une option du graphe :

```
vminor=1
```

- Ajouter cette option

10 Aspect des bulles

Nous contrôlons la couleur et la taille des bulles par l'option du graphe :

```
bcolor=red bsize=12
```

- Entrer différentes tailles pour les bulles

11 Ajout de label sur les bulles

Pour avoir de manière explicite le nombre d'ingénieur à côté de chaque bulle, nous rajoutons :

```
blabel bfont=swissi
```

- Entrer cette option avec puis sans changement de fonte

Facultatif : aller voir le code source complet dans le programme GPLBUBL1-Generating a Simple Bubble **Plot situé** dans les exemples de GPLOT : Sommaire-> Learning to use SAS ->Sample SAS programs.

12 L'instruction PLOT

L'instruction 'PLOT y*x' dans une procédure proc gplot trace le graphe de la fonction y en fonction de x. Le programme suivant affiche le graphe des maxima annuels du Dow Jones :

```
data stocks;
input year hdate :date9. high ldate :date9. low;
datalines;
1955  30DEC1955  488.40  17JAN1955  388.20
1956  06APR1956  521.05  23JAN1956  462.35
1957  12JUL1957  520.77  22OCT1957  419.79
;
proc gplot data=stocks;
plot high*year; run;
```

- Entrer ces intructions
- Reprener l'ensemble des valeurs dans le programme GPLDTPT1-Connecting Plot Data Points dans les exemples de GPLOT : Sommaire -> Learning to use SAS ->Sample SAS programs.

13 L'instruction SYMBOL

La manière de tracer le graphe est régie par l'instruction SYMBOL. On peut spécifier l'aspect des points, leur couleur et leur taille, ainsi que la maière dont les points sont reliés les uns aux autres. Par exemple changeons la couleur, la forme et la taille des points :

```
symbol1 color=red value=dot height=3;
```

- Entrer cette instruction.

Comme pour les axes, on peut définir jusqu'à quatre-vingt-dix-neuf symboles avec symbol1,..., symbol99. Les différent tracés demandés utiliseront les symboles definis par ordre croissant.

14 Connecter les points d'un graphe

Pour connecter les points d'un graphe, on spécifie dans l'instruction symbol :

```
symbol interpol=join;
```

L'instruction symbol est équivalente à symbol1.

15 Plusieurs tracés sur un même graphe avec OVERLAY

Nous pouvons placer plusieurs tracés sur le même graphe en spécifiant l'option overlay :

```
plot high*year low*year / overlay;
```

- Effectuer le graphe avec les deux tracés
- changer l'aspect du deuxième tracé avec

```
symbol2 font=marker value=C color=blue interpol=join height=2;
```

16 Lignes horizontales de références

Pour faciliter la lecture du graphe, nous ajoutons des lignes horizontales de référence avec l'option :

```
vref=1000 to 5000 by 1000 lvref=2
```

L'option lvref permet de spécifier le style des lignes de référence.

- Tester différent style pour les lignes de référence.

17 Gestion de la légende

Par défaut nous n'avons pas de légende pour le graphe. Pour en ajouter une il suffit de spécifier l'option 'legend' pour le graphe.
Nous pouvons également définir un type de légende et l'utiliser dans le graphe. La définition d'une légende est similaire à la définition d'un axe ou d'un symbole :

```
legend1 label=none shape=symbol(4,2) position=(top center inside)
mode=share;
```

Nous utilisons cette légende en entrant l'option

```
legend=legend1
```

- Mettre cette nouvelle légende.
- Enlever et manipuler chacune des options de la légende pour en comprendre le fonctionnement.

L'option shape permet de régler l'espacement entre les symboles puis leur taille.

18 Remplir de couleurs les zones délimités par les courbes

Les courbes définissent différentes zones que l'on peut colorier. Nous ajoutons à plot l'option

```
areas=2
```

Ceci indique que SAS doit colorier les deux premières zones du graphe

- Ajouter cette option. Essayer différentes valeurs.
- Simplifier les symboles en utilisant seulement

```
symbol1 interpol=join;
```

19 Plusieurs courbes indexées par une troisième variable

On peut tracer plusieurs courbes, une pour chaque valeur d'une troisième variable :

```
data citytemp;
input  month faren city $;
datalines;
   1      40.5    Raleigh
   1      12.2    Minn
   2      42.2    Raleigh
   2      16.5    Minn
;
proc gplot data= citytemp;
plot faren*month=city; run;
```

Cette fois le graphe possède automatiquement une légende.

- Entrer ces instructions. Récuperer l'ensemble des données sur l'aide de SAS :GPLVRBL2-Plotting Three Variables dans les exemples pour GPLOT.
- Régler l'épaisseur du cadre. Régler l'offset pour l'axe des abscisses.
- Définir trois symboles distincts pour les trois tracés. Changer les couleurs et le type des points.
- Changer l'interpolation et l'épaisseur des courbes avec

```
interpol=spline width=2
```

- Pour l'axe des abscisses, nous remplaçons les chiffres par les abrévations des mois lors de la définition de l'axe :

```
value=('JAN' 'FEB' 'MAR' 'APR' 'MAY' 'JUN'
            'JUL' 'AUG' 'SEP' 'OCT' 'NOV' 'DEC')
```

- Pour avoir Minneapolis écrit en entier dans la légende, nous entrons :

```
legend1 label=none value=(tick=1 'Minneapolis');
```

L'instruction 'tick=1' indique que nous nous occupons de la première entrée de la légende.

20 Un deuxième tracé avec son axe vertical avec PLOT2

L'instruction PLOT2 place sur le même graphe un deuxième tracé. Ce tracé a sa propre échelle verticale, indiquée sur un axe vertical à droite.
La variable horizontale doit être la même pour les deux tracés.

18 SAS INSIGHT

1 Création du tableau de données

Nous créons un tableau aléatoire de 20 observations :

```
data donnees;
do X=1 to 20;
Y=rand('binomial',0.5,25);
Z=X+0.4*sqrt(Y);
output;end;run;
```

2 Présentation

Le module Insight de SAS permet un accès interactif aux données. Pour le lancer, vous devez aller dans Solutions → Analyse → Analyse Interactive des données. Ouvrez le tableau créé précédemment. Vous pouvez alors éditer directement votre tableau.

3 Nouvelles variables : Edit → Variables

Nous pouvons créer de nouvelles variables à partir de celles existantes, avec les fonctions log, exp, ...
Si nous voulons la racine carrée de Z :

- soit nous cliquons sur Y puis allons chercher la fonction racine carrée dans le menu Edit → Variables
- soit aucune variable n'est sélectionnée. Nous cliquons sur racine carrée comme précédemment. Un menu apparaît alors. Sélectionner la variable Z pour le champs Y et cliquez sur OK.

4 Sauvegarde du tableau et modification

Aller dans File → Save → Data.
Nous allons ajouter une variable à l'aide de l'instruction

```
T=rand('bernoulli',0.4);
```

Pour cela il faudra fermer notre tableau interactif pour pouvoir le modifier à l'aide d'un programme. Retourner ensuite dans SAS Insight.

5 Histogrammes : Analyse → Histogram

- Dans le menu qui apparaît, demander un histogramme sur la variable Y en groupant par rapport à la variable T. Vous verrez alors deux histogrammes, un pour chaque valeur de T.
- Cliquez sur les rectangles des histogrammes. Les observations correspondantes sont alors mises en surbrillance.
- Sélectionner des observations. Leurs positions dans les histogrammes sont alors mises en évidence.
- En double-cliquant sur un rectangle, vous avez accès aux différentes valeurs des observations.
- Pour régler l'amplitude des classes, cliquez droit sur l'histogramme, sélectionnez Ticks puis donner une autre valeur pour Tick.

6 Box plot

La BoxPlot ou boîte à moustache représente sur le même graphique les valeurs extrêmes, la médiane et les quartiles.

- Effectuer la BoxPlot de la variable Y
- Mettre la variable T dans le champs Group, puis demander une nouvelle Boxplot en mettant T dans le champ X.

Le champ Group correspond à la commande BY. Le champ X crée une fonction qui associe à toute modalité (ici de T) une boîte à moustache.

- Double-cliquez sur les différentes zones de la boîte à moustache pour obtenir des informations sur les observations correspondantes.

Les points suivants sont optionnels. Ne les faites que si vous êtes très en avance.

7 Travaux complémentaires

Effectuer l'histogramme de X, la BoxPlot de Z. En cliquant sur les différents rectangles dans un histogramme, vous voyez les positions des observations correspondantes dans l'autre histogramme. Cela peut donner une idée du comportement relatif de ces deux variables.

8 Tracer des courbes

Pour tracer Z en fonction de X, utiliser Analyse→Line Plot XY. Dans le champ Y, entrez Z, et dans le champ X, entrez X. Vous obtenez alors une courbe donnant Z en fonction de X.

- Vous pouvez supprimer des observations pour le tracé du graphe en cliquant sur le carré noir à gauche d'une observation dans le tableau, puis en désélectionnant « Show in Graphs ».
- Pour faire ressortir certaines observations, sélectionnez cette fois « Label in Plots »

9 Nuage de points

Pour tracer un nuage de points il faut utiliser Scatter Plot dans Analyse.

10 Nuage de points de 3D

Nous pouvons visualiser le nuage de points associés à X, Y, Z grâce à Rotating Plot. Les commandes sur la gauche permettent d'effectuer des rotations sur ce nuage de points en 3D. Le curdeur situé à gauche règle la vitesse des rotations : s'il est placé en haut les rotations sont grandes.

- Cliquez en même temps sur l'un des histogrammes pour voir apparaître les points correspondants dans le nuage de points 3D.

11 Distribution d'une variable

L'outil Distribution (Y) de Analyse permet d'obtenir des informations sur la distribution d'une variable. On obtient ainsi un histogramme, la boîte à moustache, mais aussi des estimés et intervalles de confiance sur la moyenne, l'écart-type et la variance.

12 Régression

La régression linéaire permet de connaître le lien affine entre deux variables. Dans le tableau que nous avons créé, il y a une certaine dépendance affine entre Z et X. Etudier cette dépendance avec Fit(Y,X)

19 LECTURES SUPPLÉMENTAIRES

1 SAS sur le Web

•En premier lieu vous avez l'aide de SAS accessible en ligne :
http://support.sas.com/onlinedoc/913/docMainpage.jsp
accessible par google en tapant "sas online doc".
•Un cours très fourni sur le site polymorphe :
http://www.polymorphe.org/index.php?/Voir-details/13-SAS-cours-IUT-STID
accessible en recherchant "sas" sur le site de polymorphe.

2 SAS et Excel

•Pour importer en tableau Excel (ou autres), allez dans fichier->importer données puis suivez les indications

•Pour exporter un tableau SAS dans un fichier Excel, allez dans fichier->exporter données, sélectionnez le tableau que vous souhaitez exporter puis continuez à suivre les instructions.

3 Récupérer un tableau SAS

Je présente trois solutions pour récupérer un tableau SAS existant. Je suppose que ce tableau s'appelle "enfants508", et qu'il est situé sur "C:/Users/Etudiants/Bureau".

1 Avec libname

```
Libname td "c:/Users/Etudiants/Bureau";
data enfants;
set td.enfants508;run;
```

2 Avec l'explorateur

Placez-vous dans "bibliothèque", faites un clic droit puis nouveau. Entrez "td" pour le nom de la bibliothèque, puis "C:/Users/Etudiants/Bureau" pour l'emplacement. Avec toujours l'explorateur, placez-vous dans "Td" et copiez-collez le tableau dans Work, en changeant le nom au besoin.

3 Un double-clic

Si votre fichier SAS est associé automatiquement au logiciel SAS, vous pouvez double-cliquez dessus. Ensuite fermez le tableau ouvert, allez dans la bibliothèque temporaire qui vient d'être créée (et nommée tmp(n+1)) puis copiez-collez le tableau dans Work, en changeant le nom au besoin.

4 Les titres

1 Description

La commande title permet de définir un titre :

```
title 'Mon titre';
proc print;
run;
```

2 De Title1 à Title10

Il est possible de définir 10 lignes de titre avec les commandes Title1, Title2, ..., Title10. La commande Title est équivalente à Title1. Essayer les commandes suivantes :

```
title 'Mon titre';
title2 'titre2';
title4 'titre4';
title10 'titre10';
proc print; run;
```

3 Supprimer des titres

Une commande Title vide supprime tous les titres situés à ce niveau ou en-dessous. Ainsi le commande 'Title4;' supprime les titres 4, 5, ..., et 10 :

```
title 'Mon titre';
title2 'titre2';
title4 'titre4';
title10 'titre10';
title4;
```

Pour supprimer tous les titres, il suffit donc de taper 'Title;'.

4 Ordre des titres

En réalité, le comportement précédent ne s'applique pas seulement lorsque l'on définit un titre vide, mais à chaque fois qu'on définit un titre :

```
title 'Mon titre';
title2 'titre2';
title4 'titre4';
title10 'title10';
title4 'essai';
```

La dernière commande non seulement redéfinit le titre 4, mais supprime également le titre 10.

20 ANNEXES

1 Recommandation pour le cours

1 En cas d'erreurs

Si le programme ne marche pas, si vous avez l'impression que rien n'a changé sur la sortie, il faut regarder en détail le journal, en particulier chercher des lignes rouges indiquant des erreurs.

S'il y a des lignes rouges, remonter pour vérifier s'il n'y en a pas d'autres auparavant. Il faut toujours corriger la première erreur.

S'il y a des lignes vertes, il y a très certainement une ligne rouge au-dessus.

2 Options d'affichage

Utiliser toujours l'option :

```
option formdlim='*';
```

Si la sortie est coupée de manière intempestive, utiliser :

```
option ps=100;
```

3 Notes de cours

Il est pratique d'utiliser un document openoffice ou word pour le suivi du cours, dans lequel vous collerez les instructions entrées dans le logiciel SAS. Vous pourrez ainsi mettre facilement des commentaires sur vos programmes et également retrouver facilement les programmes des sessions précédentes.

4 Récupérer SAS

Vous pouvez récupérer une copie de SAS.

Sur un moteur de recherche, tapez "licence sas etudiant". Remplissez le formulaire et signez-le.

Rendez-vous ensuite muni de quatre DVDs vierges auprès de Joceline Goulard, bâtiment BSL bureau 309.

5 Si vous êtes en avance

Naviguez dans l'aide de SAS pour comprendre son organisation.

Essayez des programmes exemples.

Etudiez en détail avec l'aide de SAS certaines procédures :

- Proc EXPORT
- Proc DATASETS

2 Aide de SAS

L'aide de SAS est une vrai mine d'or. Il y a de nombreux programmes exemples et la description détaillée de chaque procédure. Sur un moteur de recherche, taper « SAS DOC 9.4 » puis cliquer sur « Bookshelf ».

3 Quelques règles pour les épreuves

1 Ajouter des labels à un tableau

On ne demande pas de faire un Proc Print label; Labels ... :

```
Proc print label;
label note1='note du premier examen' note2='note
du deuxième examen';
run;
```

mais de définir les labels dans une nouvelle étape Data :

```
Data Tab;
set Tab;
label note1='note du premier examen' note2='note
du deuxième examen';
```

2 La commande BY

A chaque fois que vous utilisez la commande BY, dans un Print ou un Merge par exemple, vous devez trier le tableau auparavant (sauf s'il est parfaitement clair qu'il est déjà trié, mais dans le doute ...).

3 Séparer un tableau

Lorsqu'il s'agit de séparer un tableau en plusieurs sous-tableaux, vous ne devez pas utiliser l'instruction IF :

```
data petits moyens grands;
Set enfants;
if(taille<150) then output petits;
if(150<=taille<160) then output moyens;
if(taille>=160) then output grands;
```

A la place vous devez impérativement utiliser l'instruction SELECT :

```
data petits moyens grands;
set enfants;
select;
when(taille<150) output petits;
when(taille<160) output moyens;
otherwise output grands; end ;
```