SAS 9.4 sous linux
Logiciel hermétique pour système ouvert

PHILIPPE BESSE

4ème année GMM - MMS

Attention ce cours est dense, la lecture de ce document ne suffira pas à la bonne compréhension des outils qui y sont décrits de façon synthétique. La participation active aux TDs est indispensables à l'acquisition des compétences incontournables pour une utilisation raisonnable et raisonnée de ce logiciel fort complexe.

Équipe de Statistique et Probabilités
Institut de Mathématiques de Toulouse — UMR CNRS C5219

Département Génie Mathématique et Modélisation
Institut National des Sciences Appliquées de Toulouse — 31077 – Toulouse cedex 4.

Introduction au logiciel SAS

Résumé

Ces vignettes proposent une introduction élémentaire à un usage classique du logiciel SAS pour lancer des analyses statistiques. L'objectif est volontairement restreint aux commandes et procédures de base disponibles dans la version de la licence académique de SAS correspondant également aux usages les plus fréquents dans les grandes entreprises. Sont donc concernés : SAS/Stat, SAS/Graph et l'Analyse interactive de données.

Plan du cours :

- *Introduction*
- *Gestion des données et procédures élémentaires*
- *Graphes haute résolution*
- *Macros-commandes*
- *Bases de données*

Avant propos

Le système SAS est, sinon le logiciel de traitement de données le plus complet ni le plus répandu, celui qui traite quotidiennement le plus gros volume de données. Il a acquis, depuis sa mise en route au début des années 60, une situation dominante dans beaucoup de secteurs d'activités. En France, les grandes entreprises de l'énergie et administrations : INSEE, EDF, GDF,..., toute l'industrie pharmaceutique l'ont adopté ainsi que les entreprises du tertiaire impliqués dans la gestion volumineuse de bases clientèles (banques, assurances, marketing, VPC...). Afin d'afficher une diversification de ses activités, depuis de nombreuses années, SAS ne signifie plus *Statistical Analysis Sytem* ; le calcul statistique est devenu accessoire au regard des tâches d'ingénierie globale des systèmes d'information.

Historiquement, SAS a suivi l'expansion des sites IBM sur lesquels il a été conçu et conserve, de cet environnement initial, les caractéristiques fondamentales : complexité, lourdeur, coût mais aussi puissance et efficacité. SAS Institute, a en effet adopté la stratégie d'IBM pour fidéliser ses clients et s'attache à suivre le grand principe de la compatibilité verticale ; toute nouvelle version du logiciel est censée accepter les applicatifs conçus avec les versions antérieures. Cela semble à première vue positif pour les utilisateurs mais, en un demi-siècle de développement, ceci a aussi des conséquences très néfastes sur la clarté, la cohérence, la souplesse du ou plutôt des langages spécifiques à ce logiciel. Son apprentissage est donc long, fastidieux si l'on veut en maîtriser les subtilités. Son utilisation ne se justifie que dans un environnement où les besoins en statistiques et le volume des données traitées sont considérables et qui dispose de ressources financières conséquentes : outre le coût de location du logiciel, il faut prendre en compte les salaires des statisticiens professionnels et spécialistes qui sauront développer et maintenir les applications spécifiques nécessaires aux différents services de l'entreprise. Ce sont les raisons qui font que de nombreux services de Recherche et Développement lui préfère un outil de type libre accès comme R.

1 Introduction

Le système SAS est un ensemble de modules logiciels pour la gestion et le traitement statistique des données. A travers différents types d'interfaces utilisateur. Il permet l'écriture de *Programmes SAS* qui exécutent :

- les saisies, importations, interrogations, manipulations, fusions, transformations de données ;
- les éditions d'états, tableaux de bord, de rapports, numériques et graphiques ;
- les analyses statistiques, modélisation, prévision ;
- des applications spécifiques définies sous forme de macro-commandes pouvant être pilotées par menus ou à partir d'un navigateur ;
- les éditions plus ou moins automatisées de rapports et pages web.

Depuis la version 8, SAS propose des *solutions* : *analyse guidée des données*, *analyse marketing*, *Prévision de séries chronologiques*... qui sont autant d'environnements de travail associés à une interface graphique spécifique et à une problématique. Ils permettent un traitement de l'information sans écrire une ligne de programme. Les modules *Insight* (Analyse interactive des données) et *Enterprise Miner* sont très élaborés en ce sens. Il serait certes possible, en première approche, de se contenter de cette utilisation élémentaire mais l'usage montre que ces solutions sont nécessairement limitées et qu'un usage pro-

fessionnel, associé à des contraintes spécifiques, rend incontournable l'usage d'une programmation basique.

1.1 Table SAS

Après saisie ou importation en provenance de fichiers ASCII ou d'un SGBD (Système Relationnel de Gestion de Base de Données), les données sont gérées par SAS sous la forme d'un *"SAS Data Set"* nommé par la suite *"Table SAS"*.

Une *table SAS* est l'association d'un ou deux fichiers binaires contenant les données et leur descriptif :

- Nom de la table
- Commentaire ou "label"
- Date et heure de création
- Nombre d'observations

et, pour chaque variable :

- Nom abrégé
- Signification ou "label"
- Type
- Type de codage
- Longueur
- Position

1.2 Programme SAS

Un *programme SAS* est un enchaînements d'*étapes* de gestion des données (Data Step) et d'appels de *procédures*, décrivant, dans une syntaxe souvent spécifique à chaque *module*, les traitements à réaliser sous le couvert d'*options* prises par défaut ou explicitement définies. Les différentes étapes ou procédures communiquent entre elles exclusivement par l'intermédiaire de tables SAS, permanentes ou temporaires, et avec l'extérieur par des tables SAS ou des fichiers textes usuels en un format quelconque.

```
/* exemple de programme SAS */
/* Lecture, impression et tabulation de données. */
data Europe;
    infile "edc.fun.overseas";
    input date $ 1-7 dest $ 8-10 boarded 11-13;
proc print data = europe;
proc tabulate data = europe;
    class date dest;
    var boarded;
    table date, dest*boarded*sum;
run;
```

1.3 Les modules et leur documentation

Toutes les documentations en anglais ainsi que des tutoriels sont disponibles en ligne. Des items sont spécifiques à la version de SAS utilisée (9.3), au système d'exploitation et à chacun des modules offerts à la location. Ceux les plus utilisés concernés par ce cours sont : Base SAS, SAS/STAT, SAS/GRAPH.

Base SAS C'est la documentation de base et le manuel de référence pour tous les traitements de gestion des donnés : l'*étape Data*, la syntaxe de ses commandes, la gestion des tables SAS, l'éditeur de texte des programmes. Cet item contient également la description des procédures élémentaires (Procedures Guide), du macro langage pour l'écriture de macro-commandes, des outils de production automatique des rapports et graphes (ODS) en html, des requêtes SQL de bases de données, de production de documents XML...

SAS Procedures Fonctions et syntaxes de toutes les procédures à l'exception des procédures statistiques plus complexes : statistiques élémentaires, fréquences, graphiques basse résolution, impression, tri, tabulation, transposition,...

SAS/Stat User's Guide. Toutes les procédures statistiques et la grande variété de leurs options : tous les modèles de régression, les classifications, les durées de vie, la statistique non-paramétrique, les analyses multidimensionnelles.

SAS/Graph Software Détails des possibilités graphiques en haute résolution et de leurs innombrables options.

SAS/IML Le module de calcul matriciel interactif intégré est un langage interprété, comme Matlab ou R. Il traite des objets matriciels avec la syntaxe d'un langage évolué (PL1). Il est adapté à la mise en place rapide de méthodes originales construites à partir des opérateurs classiques de l'algèbre linéaire. Très rarement utilisée dans l'industrie, il n'est pas décrit dans ce document.

2 Modes d'utilisation de SAS

2.1 Sous linux

Exécution différée

C'est la meilleure façon d'exécuter des programmes opérationnels mais longs sans rester coller à son écran. Ils sont exécutés en différé par exemple avec la commande `at`. Il suffit de taper

```
sas nom_de_fichier.sas -fsdevice x11.motif &
```

où `nom_de_fichier.sas` est un fichier contenant un programme SAS ; les résultats sont alors regroupés dans le fichier `nom_de_fichier.lst` tandis que le compte rendu de l'exécution ainsi que les messages d'erreurs se trouvent dans le fichier `nom_de_fichier.log`. Il existe de nombreuses options : type de terminal, taille mémoire.... Le caractère `&` permet de reprendre la main avant la fin de l'exécution qui se déroule en arrière plan et ainsi, éventuellement, de tuer le processus `sas` en cas de problèmes.

Mode interactif

Taper simplement la commande `sas &` qui lance SAS en tâche de fond afin de pouvoir réactiver, si nécessaire, la fenêtre du process `shell` (xterm).

2.2 Sous windows

Pour l'essentiel, une fois que SAS est lancé à partir du menu des programmes ou à partir de son icône, le fonctionnement est le même qu'en mode interactif sous Linux. L'arborescence des répertoires est évidemment spécifique au système d'exploitation mais aussi aux paramètres de configuration de l'installation. Des facilités sont apportées par Windows comme le copier / coller des graphiques entre les fenêtres SAS et un traitement de texte.

2.3 Les fenêtres

Cinq fenêtres apparaissent alors à l'écran avec une ergonomie relativement intuitive mais, très sophistiquée et ouvrant sur de très nombreuses possibilités, il serait inefficace de vouloir la décrire de façon exhaustive. Elle est à découvrir en fonction des besoins.

Chaque fenêtre contient une barre de menus déroulants contextuels :

Fichier pour lire ou écrire dans des fichiers extérieurs à SAS, importer ou exporter des données dans différents formats, quitter SAS (exit) en fermant toutes les fenêtres.

Édition pour gérer le texte (sélectionner, couper, copier, coller...),

Affichage pour rendre active une des fenêtres.

Outils pour accéder à des utilitaires de gestion de graphiques, de tables sas, de rapports, d'images, de textes et pour configurer les options personnalisant son environnement (couleurs, polices...).

Solutions pour exécuter les modules spécifiques (s'ils ont été payés !) pour la réalisation de tableaux de bord, le développement d'applications.

Fenêtre Pour changer l'organisation des fenêtre ou en sélectionner une spécifique.

Aide pour accéder à l'aide en ligne détaillée ainsi qu'à un tutoriel (*Getting started with SAS Software*).

Ces fenêtres sont :

Éditeur est un éditeur de texte rudimentaire [1], pour entrer et modifier les programmes SAS avant d'en demander l'exécution. Outre les commandes du choix "Édition" du menu, il faut savoir que pour :

Sortie affiche tous les résultats (texte) produits par l'exécution des différentes procédures. Les graphiques haute résolution apparaissent dans une fenêtre spécifique.

Journal affiche le compte rendu de la bonne exécution et les messages d'erreur. Elle est la première fenêtre à consulter.

1. Il est vivement recommander d'ouvrir un éditeur fiable et entrer le programme avant de la soumettre par simple copie de la souris. Cela permet d'éviter des mauvaises surprises : caractères spéciaux cachés, crash de SAS...

Explorateur affichage arborescent des librairies (répertoires) et tables gérées par SAS.

Résultas permet de gérer l'ensemble des résultats (textes et graphiques) de façon arborescente.

- insérer n lignes il faut taper `in` dans la zone des numéros de lignes,
- supprimer une ligne il faut y taper `d` et, pour supprimer un block, entrer `dd` sur la première et la dernière ligne du bloc,
- passer alternativement du mode insertion au mode superposition il faut taper `<ctrl>x`.
- Enfin, le menu spécifique **Exécuter** permet de lancer l'exécution (comme $< F3 >$) du programme de l'éditeur ou celui du tampon copié avec la souris et de rappeler (comme $< F4 >$) le programme exécuter dans la fenêtre
-

D'autres fenêtres s'ouvrent par exemple lors de la production de graphes ou au lancement de modules interactifs spécifiques.

Attention aux courants d'air, il est important de gérer correctement la multitude des fenêtres qui remplissent l'écran et surtout de les refermer dans le bon ordre afin d'éviter de se retrouver bloqué, par exemple sur un *popup* qui attend une réponse.

2.4 Bibliothèques

Ce sont, du point de vue du système d'exploitation, les répertoires dans lesquels SAS gère les fichiers et tables SAS de façon temporaire, le temps d'une session ou exécution du programme, ou permanente.

SasUser : bibliothèque permanente créée par défaut. Les tables de cette bibliothèque sont nommées `sasuser.nomtab`.

Work : bibliothèque contenant les tables temporaires créées par défaut par les différentes étapes et procédures. Elles sont effacées à la fin de la session ou de l'exécution du programme. Les tables sont nommées `work.nomtab` ou plus simplement de façon implicite : `nomtab`.

SasHelp contient un groupe de catalogue permettant le fonctionnement par défaut d'une session SAS, les données servant d'exemples et toute l'aide en ligne.

Maps contient les tables de données géographiques (fonds de carte).

Plutôt que d'utiliser toujours la librairie `SasUser`, il est possible de définir sa propre librairie (ou répertoire) de tables permanentes (commande `libname`).

3 Commandes globales

Cette section aborde quelques éléments de programmation SAS. En plus des deux types de structures déjà cités (étape data et procédures), certaines commandes peuvent intervenir à tout moment afin de préciser ou modifier certaines options. Elles restent actives durant toute la session ; voici les principales.

3.1 Référencement

libname `libref 'SAS-data-library';`
Les répertoires (unix ou windows) contenant des bibliothèques de tables SAS autres que `sasuser` (par défaut) doivent être alloués avant leur utilisation mentionnant cette référence logique.

filename `fileref 'external-file';`
Un fichier extérieur à SAS (fichier ASCII) peut également être référencé.

3.2 Mises en formes des sorties

Commentaires Tout texte sous la forme : `* texte ;` ou `/* texte */` est ignoré de SAS et permet d'expliciter le fonctionnement d'un programme.

options `liste d'options;`
permet de spécifier certains paramètres :

- *obs*=n limite le nombre d'observations à traiter pour tester par exemple un programme,
- *linesize*=n fixe le nombre de caractères par ligne en sortie,
- *pagesize*=n fixe le nombre de lignes par page,
- *date*|*nodate* présence ou absence de date en haut de page,
- *number*|*nonumber* présence ou absence de numérotation des pages,
- *pageno*=n fixe le numéro de la première page.

title$n$ `'titre';`
permet de faire imprimer jusqu'à $n = 10$ lignes de titre sur chaque page en sortie, par défaut, la première ligne contient : `The SAS System`,

footnote$n$ `'note de bas de page';`
permet de faire imprimer jusqu'à $n = 10$ notes de bas de page sur chaque page en sortie.

Options usuelles :

```
options linesize=75 pagesize=66 nodate number;
title "Mon titre";
footnote "Note de bas de page";
```

3.3 Production de rapports

Il est possible depuis la version 8 de produire directement des sorties dans un format standard afin de les introduire sans un traitement de texte (`.rtf`) ou une page web (`.html`). C'est le système ODS (*open delivery system*).

```
ods rtf body="nomfichier.rtf";
/* Programme SAS */
ods rtf close;
```

Remplacer `rtf` par `html, ps, pdf` fournit d'autres formats de sortie.

La version 9 permet également la sortie de graphiques sur le même principe. Ils sont de bonne définition et directement produits pas les procédures statistiques (`princomp, anova, corr, reg, glm`...) mais leurs options sont contraintes. Ces possibilités seront abordées au momment de l'utilisation de ces procédures.

4 Analyse interactive des données

A notre grand regret, le module SAS/Insight d'analyse interactive des données n'est plus maintenu dans SAS à partir de la version 9.4. L'introduction à ce module pour très pratique et très efficace pour faire de l'exploration efficace de données est en conséquence supprimée. On peut penser que la stratégie de SAS consiste à favoriser le développement d'un autre produit équivalent : JMP mais qui est plus compliqué à mettre en œuvre et puis c'est un autre logiciel ! Encore une raison pour utiliser plus systématiquement l'open source R.

SAS de base : gestion des données et procédures élémentaires

Résumé

Description des commandes (module SAS de base) les plus utiles de l'étape `data` *et principales procédures élémentaires pour créer, transformer, normaliser, concaténer, fusionner, transposer, trier des tables SAS. Procédures de calcul des statistiques élémentaires.*

Retour au plan du cours.

1 Introduction

Le système SAS offre un très large éventail de traitements permettant d'assurer toutes les fonctions de gestion des données qu'un statisticien peut être amené à mettre en œuvre. Cette richesse est aussi source de complexité. Nous nous contenterons de décrire les exemples les plus standards. Ces traitements sont pris en charge par les étapes `data` (*data step*), qui reconnaissent un langage de programmation proche de PL/1 [1], et quelques procédures spécifiques.

Un *programme SAS* se décompose en :

1. une (ou des) étape(s) `data`,
2. une (ou des) appels à des procédures (`proc`).

Ces deux parties sont généralement indépendantes, il est donc inutilement coûteux de ré-exécuter l'étape data, dont les résultats sont stockés dans une table SAS (permanente ou temporaire), à chaque mise au point des procédures qui suivent.

Les commandes élémentaires :

```
proc print;
run;
```

1. Créé par IBM dans les années 70s dans l'espoir de remplacer à la fois le Fortran et le CobolL ; ce fut le premier langage structuré. Le déploiement conjoint d'Unix et du langage C a été fatal au PL/1.

permettent de vérifier dans la fenêtre `sortie` la bonne constitution de la dernière table SAS créée.

2 Étape *Data* de création d'une table

2.1 Lecture d'une table

L'objectif est de construire une table SAS dans le cas de figure le plus usuel où les données sont contenues dans un fichier texte (ascii) provenant d'un éditeur, d'un autre logiciel statistique ou encore d'un gestionnaire de bases de données. Les données peuvent également être directement saisies au clavier (SAS/Insight) ou incluses au programme SAS (commande `include card`). Enfin, SAS peut lire tout autre format de fichier que ceux succinctement décrits ci-dessous (articles de longueur variable, binaire,. . .) sur tout support mais c'est vite compliqué !

syntaxe

data <sasuser.>table-sas ;
infile fileref ou 'nom-de-fichier' <dlm= 'carac' lrecl=nn> ;
input liste de variables et spécifications ;

Options

La liste des variables définit chaque identificateur ; il est suivi du caractère "$" pour préciser, éventuellement le type alphanumérique de la variable. Il est *fortement* conseillé de déclarer chaque variable qualitative de type alphanumérique.

- *dlm=';'* si le délimiteur entre les valeurs est un ';' ou '`09x`' pour un caractère de tabulation,
- *dsd* deux délimiteurs successifs sont interprétés comme une valeur manquante, sinon insérer un '.',
- *lrecl* si la ligne est très longue, majorant du nombre de caractères d'un enregistrement,
- *firstobs* numéro de la ligne à laquelle commencer la lecture,
- *obs* numéro de la dernière ligne à lire.

2.2 Lecture en format libre

Les données associées à une unité statistique sont séparées par des blancs ou tout autre caractère délimiteur spécifié dans l'option `dlm(;:,...)` ; les unités sont séparées par des retours à la ligne. Si les lignes du fichier en entrée sont trop longues, dépassant la valeur par défaut, elles sont tronquées en lecture. Il est nécessaire de déclarer un majorant de cette longueur dans l'option `lrecl`.

Une observation par ligne

Chaque observation est décrite par une ligne du fichier.

```
data sasuser.fich1;
    infile '~/data/fich.dat' dlm=';';
    input var1 var2 $ var3;
run;
```

Attention, si une donnée manque dans le ficher pour satisfaire le nombre de variables, SAS poursuit la lecture sur la ligne suivante sans prévenir. Il est primordial de toujours vérifier la bonne lecture des données. Le cas échéant, une donnée manquante est codée par un point ".".

Une observation sur plusieurs lignes

Chaque unité statistique est décrite sur n lignes du fichier. Plusieurs déclarations `input` ou des caractères "/" permettent de préciser le découpage.

```
data sasuser.fich1;
    infile '~/data/fich.dat' dlm=';';
    input var1 var2 $ var3 /
          var4 $ var5 var6;
run;
```

Plusieurs observations par ligne

Le double caractère "@@" a pour effet de maintenir un article dans le buffer de lecture jusqu'à ce qu'il soit complètement lu.

```
data sasuser.fich1;
    infile '~/data/fich.dat';
    input var1 var2 $ var3 @@ ;
run;
```

Lorsque le volume des données est très réduit, celles-ci peuvent être intégrées au programme avec la commande `cards` :

```
data sasuser.fich1;
    input var1 var2 $ var3 @@ ;
    cards;
1.5 A 55 2.4 B 44 2.7 B 61 2.3 A 48
;
run;
```

Attention à la place du dernier point-virgule.

2.3 Lecture formatée

Le format libre ci-dessus n'est plus utilisable lorsque, pour des raisons d'économies d'espace disque, les données sont collées ; il faut alors indiquer explicitement, à la suite du nom de chaque variable, les positions (ou champs) concernés.

```
data sasuser.fich1;
    infile '~/data/fich.dat';
    input var1 1-12 var2 $ 13-18 var3 19-25;
run;
```

2.4 Importation / exportation

Les données peuvent être directement importées à partir d'autres formats car issus d'autres logiciels comme Excel si le module SAS/Access est implémenté pour pouvoir accéder à ce type de format. Sinon seuls certains type de fichier sont accessibles comme ceux `.csv`. C'est le rôle de la procédure `import` dont voici un exemple d'utilisation. Le fichier `.csv` contient une première ligne avec les noms des variables, les valeurs sont séparées par des "," et la marque décimale est le point. De manière réciproque, une table SAS peut être exportée dans un format donnée, texte (ascii) ou Excel.

```
proc import datafile="nom-de-fichier.csv"
```

```
      out=sasuser.table-sas dbms=dlm replace;
getname=yes;
datarow=2;
run;
/* traitements*/
proc export data= sasuser.table-sas
     outfile="nom-de-fichier2.xls"
     dbms=excel;
```

De façon plus élaborée, la procédure SQL autorise l'interrogation de bases de données (Oracle, Access, MySQL...) en relation avec le module SAS/Access.

3 Étape *data* de transformations

3.1 Fonctionnement

L'étape data est capable d'interpréter un langage de programmation évolué qui, par sa syntaxe, est proche de PL/1. On y retrouve les mêmes structures : `if, then, else, do`. La différence fondamentale est qu'une étape data peut être assimilée, en général, à une lecture de la table à traiter. Elle inclut implicitement une boucle considérant chacune des observations de la première à la dernière d'une table ; une "variable" du langage est une colonne ou variable statistique.

Syntaxe

La syntaxe habituelle est la suivante :

data <sasuser.>table_out ;

set <sasuser.>table_in ;

... *instructions* ;

run ;

Chaque observation ou ligne de `table_in` est lue, transformée par exécution des `instructions` puis enregistrée sur `table_out`. Par défaut, toutes les variables de `table_in` sont considérées et recopiées sur `table_out` ainsi que celles qui ont été créées par les `instructions` mais il est possible d'en laisser tomber (`drop`) ou de n'en conserver (`keep`) que certaines.

Keep *et* Drop

Ces commandes peuvent apparaître comme des options des commandes `data`, `set` ou d'autres procédures :

data `table_out (drop=var1 var2);`

set `table_in (keep=var1 var2 var3);`

ou encore comme commandes d'une étape data :

`keep|drop var1 var2`

L'utilisation de `drop` ou `keep` dépend du nombre relatif de variables à éliminer par rapport au nombre à conserver.

3.2 Fonctions

Le langage reconnaît les expressions arithmétiques usuelles, sait gérer des constantes numériques ou alphanumériques (`'constante'`) et des "variables" qui sont enregistrées sur `table_out` à moins d'avoir été éliminées (`drop, keep`). Il reconnaît, de plus, la plupart des fonctions mathématiques usuelles (`round, sin, log, sqrt,...`), les fonctions de gestion de chaînes de caractères (`lenght, scan, substr,...`), celles spécifiques aux différentes lois de probabilités (quantiles) et d'autres à usage plus statistique (`sum, mean, min, max, var, std,...`). Ces dernières s'appliquent à une liste de valeurs avec la syntaxe suivante :

sum `(var1,` **of** `var10-var20, var 25)`

3.3 Exemples

```
/* transformations de variables quantitatives */
data sasuser.table1;
    set sasuser.table2 (drop=var10 var11);
    newvar1=sqrt(var3);
    newvar2=log(var4)/mean(of var5-var9);
run;

/* codage en classes d'une variable quantitative */
data sasuser.table1 (keep taillec sexe csp);
  set sasuser.table2 ;
  if sexe='M' then do;
```

```
    if taille > 190 then  taillec='grand';
      else if taille > 170 then taillec='moyen';
                           else taillec='petit';
                end;
    else then do;
if taille > 180 then  taillec='grand';
      else if taille > 160 then taillec='moyen';
                           else taillec='petit';
                end;
run;

/* codage en classes d'une variable quantitative */
/* meme chose avec la commande select :*/
data sasuser.table1 (keep taillec sexe csp);
  set sasuser.table2 ;
  select (sexe);
  when ('M')
if taille > 190 then  taillec='grand';
      else if taille > 170 then taillec='moyen';
                           else taillec='petit';
  when('F')
  if taille > 180 then  taillec='grand';
      else if taille > 160 then taillec='moyen';
                            lse taillec='petit';
  otherwise put 'probleme';
  end;
run;
```

Il est facile de supprimer ou plutôt, de ne pas recopier dans la table créée des observations vérifiant ou non une condition logique.

```
/* exemples de regroupement de modalités */
select(mon);
when(mon in('mars','avri','mai ')) sais='printps';
when(mon in('juin','juil','aout')) sais='ete';
when(mon in('sept','octo','nove')) sais='automne';
when(mon in('dece','janv','fevr')) sais='hiver';
otherwise;
end;
select(a);
when(1);
when(3,4,5) x=x*10;
otherwise;
end;

/* élimination d'observations */
data sasuser.table1;
    set sasuser.table2 ;
    if var1 = 'nul' then delete;
/* SAS passe a la suivante*/
    ...
run;

/* selection implicite d'observations */
data sasuser.table1;
    set sasuser.table2 ;
    if var1 = 'bon' ;
/* sinon SAS passe a la suivante*/
    ...
run;
```

Des commandes spécifiques à l'étape Data n'ont pas été décrites : `retain, return, put, output, missing, list, link, label, goto, do`. Ainsi, l'exemple suivant utilise une variable définie de façon implicite et une boucle pour répéter les différents niveaux d'un facteur.

```
/* définition implicite d'une variable */
data table1; /* fichier temporaire */
    input var n ;
    do i=1 to n ;
    input gain @@ ;
    output ;
    end;
```

```
    cards ;
0     16
  28 29 26 24 26 25 229 23 29 24 20 22 20 29 22
0.04 11
  186 229 220 208 228 198 222 273 216 198 213
0.07 12
  179 193 183 180 143 204 114 188 178 134 208 196
0.10 8
  130 87  135 116 118 165 151 59
0.13 11
  154 130 130 118 118 104 112 134 98 100 104
;
run;
```

3.4 Tableau de variables

Des traitements plus sophistiqués (itératifs, conditionnels) peuvent être opérés sur les variables d'une table en les déclarant sous la forme d'un tableau (`array`) dont les colonnes peuvent alors être indicées. Plus précisément, cette déclaration revient à considérer un vecteur ligne contenant successivement chaque ligne de la table en cours de lecture.

De nombreux exemples sont fournis dans les textes des macros-commandes.

4 Procédures de transformation

D'autres transformations usuelles sont proposées sous forme de procédures.

4.1 Rangs

La procédure `ranks` calcule les rangs des valeurs de variables quantitatives et les recopie dans une nouvelle table. Par défaut, les valeurs égales sont affectées du rang moyen. Une option (`group=`) permet de spécifier le nombre de valeurs de rangs utilisées et ainsi de découper en classes une variable quantitative avec des effectifs sensiblement égaux.

Syntaxe

proc rank <options> ;
by <descending> variable ;
ranks liste de nouvelles variables ;
var liste de variables ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,
- *out=table sas* spécifie le nom de la table créée qui contiendra les variables initiales et les rangs,
- *fraction|groups=n|normal=blom* pour obtenir, respectivement, les valeurs de la fonction de répartition, un découpage en n classes de même effectif, les valeurs d'une distribution normale, plutôt que, par défaut, les valeurs des rangs.
- *descending* rangs par valeurs décroissantes,
- *ties=* spécifie la façon de gérer les ex-æquos (`mean|high|low`).

Commandes

by suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations ; la table doit être triée.

ranks doit être spécifiée si l'on veut que les variables initiales soient recopiées en sortie. Sinon, les variables gardent le même nom. Il y a une correspondance terme à terme entre les noms des deux listes de variables.

var les rangs des variables de la liste *var* sont calculés et recopiés dans la table de sortie ; par défaut, toutes les variables numériques sont traitées.

4.2 Normalisation

La réduction ou la "standardisation" de variables quantitatives s'obtient en exécutant la procédure `standard`.

Syntaxe

proc standard <options> ;
by <descending> variable ;
var liste de variables ;

weight variable ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,
- *out=table sas* spécifie le nom de la table créée qui contiendra les variables initiales et celles standardisés.
- *print* imprime moyennes et écarts-types des variables traitées,
- *mean=* spécifie la nouvelle valeur moyenne (=0),
- *std=* spécifie la nouvelle valeur de l'écart-type (=1),
- *replace* demande que toute donnée manquante soit remplacée par la nouvelle moyenne (*mean=*),
- *vardef* précise le diviseur dans le calcul de la variance (df, n, wdf,wgt).

Commandes

by suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations ; la table doit être triée.

var les variables de la liste *var* sont standardisées et recopiées dans la table de sortie ; par défaut, toutes les variables numériques sont traitées.

weight nom de la variable contenant les pondérations des observations.

4.3 Transposition

La procédure `transpose` lit toute ou partie d'une table SAS et la recopie après transposition : les lignes deviennent des colonnes et les colonnes des lignes. Une nouvelle variable `_name_` contient alors, en sortie, les noms des variables transposées qui désignent maintenant les observations. La commande `by` permet de réorganiser des données complexes. Attention, si une variable est de type caractère, toutes les variables transposées le deviennent.

Syntaxe

proc transpose <options> ;
var liste de variables ;
id variable ;
copy liste de variables ;
by liste de variables ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,
- *out=table sas* spécifie le nom de la table créée qui contiendra les variables transposées,
- *prefix=* spécifie le préfixe utilisé pour créer les noms des nouvelles variables,
- *name=* spécifie le nom de la variable créée qui contiendra les noms des anciennes variables pour désigner les observations (par défaut `_name_`).

Commandes

by : une observation est créée pour chaque variable transposée et pour chaque groupe ; la variable de groupage est incluse en sortie mais non transposée. La table doit être triée.

copy les variables de la liste *copy* sont recopiées dans la table de sortie sans transposition.

var les variables de la liste *var* sont transposées et recopiées dans la table de sortie ; par défaut, toutes les variables numériques, n'apparaissant pas ailleurs sont traitées.

id nom de la variable en entrée contenant les noms des variables après transposition.

4.4 Tri

La procédure `sort` permet de trier une table SAS.

Syntaxe

proc sort <options> ;
by <descending> variable1 <<descending> variable2> ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,
- *out=table sas* spécifie le nom de la table créée qui contiendra les observations triées.
- *nodup* élimine les observations identiques.

Commandes

by liste des variables qui servent de clé de tri ; l'ordre est croissant par défaut.

Sur les tables triées, SAS gère deux variables générées par la commande `BY` permettant d'identifier les débuts et fin de groupes de lignes prenant les mêmes valeurs d'une variable utilisée dans le tri. Par exemple, si la table est triées par la variable `sexe`, la commande `BY sexe` génère les variables temporaires `First.sexe` et `LAST.sexe` qui prennent des valeurs 0 ou 1 pour indiquer respectivement la première observation ou la dernière de chaque modalité de la variable.

5 Concaténation de tables

5.1 Concaténation verticale et fusion

Cette opération consiste à compléter une table SAS par une ou plusieurs autres contenant les mêmes variables mesurées sur d'autres observations. Elle est très simple à réaliser, il suffit de mentionner toutes ces tables dans la commande `set` en renommant, éliminant, conservant éventuellement certaines variables. S'il n'y a pas une bonne correspondance entre les variables, des données manquantes sont générées.

```
data sasuser.concvtable;
    set sasuser.table1 (rename=(var1=var15))
        sasuser.table2 (rename=(var5=var15));
run;
```

Si chacune des tables est triée sur la même clé, ce tri peut être conservé dans la table concaténée, il s'agit alors d'une **fusion**, en spécifiant la variable clé dans une commande `by`.

```
data sasuser.fusiontable;
    set sasuser.table1 (rename=(var1=var15))
        sasuser.table2 (rename=(var5=var15));
    by vartri;
run;
```

D'autres possibilités sont offertes par la procédure `append`.

5.2 Concaténation horizontale

Les mêmes unités statistiques ont été observées sur des paquets de variables contenues dans des tables SAS distinctes. La table regroupant toutes les variables est obtenue en utilisant plusieurs fois la commande `set`.

```
data sasuser.conchtable;
    set sasuser.table1;
    set sasuser.table2;
run;
```

Le même résultat peut être obtenu avec la commande `merge`. Elle permet, en plus, de contrôler la bonne correspondance des lignes de chaque table (triée) suivant les valeurs d'une clé et introduit, le cas échéant, des données manquantes.

```
data sasuser.mergetable;
    merge sasuser.table1 sasuser.table2;
    by varcom;
run;
```

5.3 Affichage d'une table

Le menu affichage permet de vérifier le bon contenu d'une table. Par ailleurs, la procédure `print` édite la liste d'une table SAS dans la fenêtre *SAS.sortie* en calculant éventuellement des sommes partielles.

Syntaxe

proc print <options> ;
by <descending> variable ;
var liste de variables ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,
- *noobs* supprime les numéros des observations,
- *round* arrondit les résultats avec deux décimales.

Commandes

by suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations ; la table doit être triée.

var les variables de la liste *var* sont éditées dans la fenêtre `sortie` ; par défaut, toutes les variables sont traitées.

6 Procédures statistiques élémentaires

6.1 procédure `tabulate`

La procédure tabulate est très intéressante à manipuler et apporte beaucoup de richesse pour les sorties SAS. Des livres entiers lui sont d'ailleurs consacrés. Elle permet en particulier de créer des *tableaux de bord* (reporting) synthétiques qui compilent, récapitulent et éventuellement analysent les données. La syntaxe est la suivante :

Syntaxe

proc tabulate <options> ;
class <liste$_1$ > ;
var <liste$_2$ > ;
table ... ;
run ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,

Chaque ligne de commande a un rôle très spécifique en fonction de la syntaxe et de la ponctuation.

- Les variables qualitatives contenues dans la liste 1 servent à définir des groupes d'observations sur lesquels des statistiques seront calculées. Ce sont ces variables qui définiront les lignes et les colonnes du tableau à calculer.
- Les variables contenues dans la liste 2 doivent nécessairement être numériques. C'est sur ces dernières que l'on pourra effectuer des opérations.
- l'instruction table permet de préciser et définir l'architecture du tableau. En particulier, il est possible de concaténer, croiser ou encore regrouper des catégories.

La maîtrise de l'instruction table est donc nécessaire si l'on souhaite construire des tableaux compliqués. Cette instruction est extrêment sensible à la ponctuation : la virgule marque la limite entre les lignes et les colonnes, un espace indique une juxtaposition de deux éléments dans une même dimension (ligne ou colonnes) et l'étoile signifie que l'on imbrique deux éléments dans une même dimension. Typiquement, on utilisera donc la syntaxe suivante :

```
table (lignes) , (colonnes * cellules);
```

Entre les parenthèses de la dimension `ligne`, on trouve des noms de variables citées dans class. Il en va de même entre les parenthèses de la dimension colonne.

Pour les cellules, on retrouve soit une statistique comme une fréquence ou un pourcentage, soit une variable de calcul (déjà citée dans var) suivie d'une étoile et de sa statistique.

```
data exemple; /* Création de la table */
   input jour veh $ effectif ener $;
   cards;
   1 voiture 6 e
   1 camion 3 g
   2 voiture 4 g
   2 camion 5 g
   3 voiture 12 e
   ;
run;
```

Exemple d'utilisation de la procédure.

```
proc tabulate data=exemple;
var jour effectif;
class veh;
table veh, effectif*max effectif*mean;
run;
proc tabulate data=exemple;
var effectif;
class veh ener;
```

```
table veh, ener*(effectif*mean);
run;
```

6.2 procédure `univariate`

Cette procédure regroupe tous les résultats qui peuvent être obtenus, par SAS, dans le cadre d'une étude uni-variée de variables quantitatives : indicateurs de tendance centrale (moyenne, médiane, mode), indicateurs de dispersion, d'autres caractéristiques de la distribution (quantiles, skewness, kurtosis), les graphiques en basse résolution (histogrammes, tige-et-feuille, boîte-à-moustaches, droites de Henri), le test de Student de nullité de la moyenne, les tests de normalité d'une distribution et les tests non-paramétriques (signe, Wilcoxon,...).

Syntaxe

proc univariate <options> ;
var liste de variables ;
by <descending> variable ;
weight variable ;
output <out=table sas> <liste de statistiques> ;

Options

La liste des options permet de préciser les résultats attendus.

- *data=table sas* indique le nom de la table par défaut, la dernière créée,
- *normal* pour obtenir des tests de normalité,
- *plot* pour obtenir les graphiques, si la commande *by* est employée, les boîtes sont affichées en parallèle,
- *vardef=* précise le diviseur dans le calcul de la variance (df, n, wdf, wgt).

Commandes

by suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations ; la table doit être triée.

output indique le nom du fichier et la liste des statistiques qui y seront enregistrées.

var liste des variables concernées par la procédure, par défaut, toutes les variables quantitatives.

weight nom de la variable contenant les pondérations des observations.

6.3 procédure `means`

Les résultats fournis par cette procédure sont inclus dans ceux produits par la procédure *univariate* décrite ci-dessus. Elle s'utilise de la même façon et diffère par la présentation des résultats résumés sous la forme d'un tableau plus facile à consulter. Elle ne fournit ni statistique non-paramétrique ni graphe. La procédure *summary* ne diffère de la procédure *means* que dans le choix des options par défaut.

6.4 procédure `freq`

Cette procédure traite les variables qualitatives. Elle fournit donc des tris à plat et complète ces résultats par des études bi ou multi-variés de tables de contingences. On obtient ainsi les profils lignes et colonnes, les statistiques des tests d'indépendance (χ^2) et des comparaisons avec les valeurs déduites du modèle d'indépendance.

Syntaxe

proc freq <options> ;
by <descending> variable ;
tables liste des croisements requis </ options> ;
weight variable ;

Options

- *data=table sas* indique le nom de la table par défaut, la dernière créée,
- *order=freq* édition ordonnée par effectifs décroissants,

Commandes

by suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations ; la table doit être triée.

tables liste des croisements exprimés sous une des formes : a*b, a*(b c), (a b)*(c d), (a- -d)*c. Les options précisent les résultats et statistiques demandées ; la plus utile est `chisq` qui exécute un test du χ^2, d'autres permettent d'éviter certaines éditions (profils).

weight nom de la variable contenant les pondérations des observations.

6.5 procédure `corr`

Cette procédure étudie les liaisons entre variables quantitatives et propose donc les indicateurs usuels comme les coefficients de corrélation de Pearson et de Spearman, d'autres qui le sont moins, et les tests associés. Les résultats peuvent être enregistrés dans des tables sas.

Syntaxe

proc corr <options> ;
by <descending> variable ;
var liste de variables ;
weight variable ;
with liste de variables ;

Options

- *data=table sas* indique le nom de la table, par défaut, la dernière créée,
- *hoeffding kendall pearson spearman* sélectionne les types de mesure de corrélation, *pearson* par défaut,
- *vardef=* précise le diviseur dans le calcul de la variance (df, n, wdf,wgt),

Commandes

by suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations ; la table doit être triée.

var les variables de la liste *var* sont croisées avec celles de la liste *with* ; par défaut, toutes les variables numériques.

with par défaut, tous les couples de variables apparaissant dans la liste var.

weight nom de la variable contenant les pondérations des observations.

6.6 Produit de matrices avec `score`

La procédure `score` permet de calculer le produit entre deux matrices (représentées par deux tables SAS) sans faire appel au module SAS/IML. La procédure `score` multiplie, individu par individu, les variables d'une table SAS par des coefficients (appelés scores) associés à ces variables et présents dans l'autre table SAS considérée. Cette procédure est utilisée, par exemple, pour calculer les coordonnées sur le plan factoriel d'un individu supplémentaire à partir des résultats d'une ACP effectuée au préalable. On peut l'appliquer directement sur les sorties des procédures `princomp`, `candisc`, `factor...`, qui contiennent une matrice de type score.

Syntaxe

proc score <options> ;

by <descending> variable ;

id variable ;

var liste de variables ;

run ;

Options

La procédure **score** multiplie les individus d'une table SAS par des "scores" contenus dans une autre table SAS. Elle effectue le produit de la matrice des "individus" par la transposée de la matrice des "scores". La table contenant les scores doit posséder une variable `_TYPE_`. Cette variable peut prendre les valeurs `SCORE` pour indiquer les coefficients, et `MEAN` et `STD` si on souhaite centrer et réduire les données avant d'appliquer les coefficients.

- *data=table* sas contenant les individus,
- *out=table* sas en sortie. Par défaut, elle contient les nouvelles variables et les variables de la table initiale.
- *score=table* sas contenant les coefficients. Elle doit posséder une variable `_TYPE_` et peut également comprendre une variable `_NAME_` donnant le nom de la nouvelle variable ou des nouvelles variables (colonnes) créées.
- *type=nom* indique que les coefficients sont conservés dans les lignes pour lesquelles `_TYPE_` prend la valeur `nom` (cf. exemple de la régression où `TYPE=PARMS`). Cette option est à utiliser si `_TYPE_` est différent de `SCORE`.
- *nostd* indique que l'on ne centre pas et que l'on ne réduit pas les données avant application des coefficients.

Commandes

by suivi du nom d'une variable qualitative pour effectuer les calculs selon les groupes définis par les modalités de cette variable. La table doit être triée.

var liste des variables sur lesquelles on applique les coefficients. Ces variables doivent être présentes dans les tables appelées par `score=` et `data=`.

id liste des variables présentes dans la table sortie définie par `out=`. Par défaut toutes les variables.

Exemples

```
/* Creation de la table contenant les scores */
data coeff;
   input _TYPE_ $ _NAME_ $ var1 var2;
   cards;
MEAN clin 1 3
STD  clin 2 1
SCORE clin 0.5 -0.5
;
run;
/* Creation des données sur lesquelles
    on souhaite appliquer les scores */
data donnees;
    input var1 var2 @@;
    cards;
3  4  1  2 .....
;
run;
/* Creation de la nouvelle variable clin
à partir de scores après centrage et
réduction des données */
proc score data=donnees score=coeff out=sortie;
    var var1 var2;
run;
```

qui donne la table `sortie` suivante :

```
  var1  var2  clin
   3     4     0
   1     2     0.5
  ..    ..    ..
```

```
/* 0 = (3-1)/2*0.5 +(4-3)/1*(-0.5) */
/* 0.5= (1-1)/2*0.5 + (2-3)/1*(-0.5) */
```

Utilisation des résultats de la procédure `reg` pour calculer la valeur ajustée de nouvelles observations.

```
proc reg data=donnees outest=coeff;
    model y = var1 var2;
run;
/* affichage de la table coeff contenant les
   coefficients de la regression */
proc print data=coeff;
run;
/* table coeff */
_TYPE_  _DEPVAR_ ... INTERCEP  var1  var2
 PARMS      y          1.5      0.3   -0.7
```

Il est alors possible, pour une nouvelle observation, d'estimer la valeur de la variable `y` en appliquant la procédure score. Celle-ci tient automatiquement compte de la constante `INTERCEP` du modèle si on précise `type=PARMS`.

```
/* calcul des y ajustés pour les nouvelles
   observations */
proc score data=autretab score=coeff type=PARMS
   out=valajust;
   var var1 var2;
run;
```

Calcul des coordonnées d'individus supplémentaires (ACP).

```
/* ACP du tableau initial */
/* la table coeff contient les coordonnées des
 vecteurs propres permettant le calcul des composantes
 principales */
proc princomp data=table1 outstat=coeff;
var var1-var5;
run;
```

```
/* Calcul des composantes principales des individus
   supplémentaires, on centre et réduit les données
 automatiquement avant calcul des composantes
 principales qui sont sauvées dans la table
 compr2 */
proc score data=table2 score=coeff out=compr2;
var var1-var5;
run;
```

Graphiques Haute Résolution avec SAS/GRAPH

Résumé

Le module **SAS/GRAPH** *permet de tracer des graphes dont la résolution est adaptée au périphérique utilisé (écran, imprimante, page web...) et avec toutes les options possibles. Cette vignette décrit les principales procédures (gchart, gplot, annotate) et options parmi un nombre de disponibles qui est considérable. Bien que la plupart des procédures statistiques fournissent (*open delevery system*) des graphiques par défaut, la construction spécifique* du *graphique adapté au problème posé est souvent incontournable.*

Retour au plan du cours.

1 Introduction

1.1 SAS/Graph

Le module SAS/GRAPH propose une très grande variétés de procédures graphiques assorties d'un nombre considérables d'options. En plus des procédures de base : gchart et gplot, bien d'autres procédures et types de graphiques sont accessibles :

- fonds de cartes (`gmap`),
- surfaces ou nuages de points en trois dimensions (`gcontour, g3d`),
- dessiner d'autres polices de caractères (`gfont`),
- combiner plusieurs graphes de natures différentes sur une même page (`greplay`),
- tester l'installation de SAS/GRAPH, les paramètres du périphérique courant et les options en vigueur (`gtestit`),
- `gradar, gkpi, gslide, gtile, gareabar, gbarline`...

D'autre part, tout texte ou figure géométrique complémentaires peuvent être rajoutés sur un graphe en les décrivant dans une table SAS spéciale dite d'annotation : *Annotate Data Set*.

1.2 ODS Graphics

Pour "simplifier" le paysage, de nouvelles procédures graphiques directement accessibles du module SAS de base et sans utiliser SAS/Graph, sont apparues à partir de la version 9.2 (en test puis en production dans la 9.3) de SAS en relation avec le service ODS : `sgdesign, sgpanle, sgplot, sgrender, sgscatter`. Les objectifs sont les mêmes mais les jeux d'options et syntaxes changent ! Néanmoins la même logique de définition des graphiques est utilisée notamment dans notamment avec les tables d'annotation.

Comme ce nouveau service ODS (*open delevery system*) *Graphics*, proposent systématiquement des graphiques par défaut dans toutes les procédures statistiques, il est légitime de s'interroger sur la nécessité de mettre en œuvre des procédures spécifiques, lourdes, complexes, notamment par la richesse des options proposées. L'expérience montre assez systématiquement, qu'utiles et efficaces en phase exploratoire ou de mise au point, des graphes par défaut sont vite limités. La compréhension d'un résultat statistique, sa portée, son impact sont souvent le résultat d'un graphique pertinent adapté à la structure des données et à la méthode utilisée. Quelque soit le logiciel, une pratique professionnelle requiert les compétences indispensables pour déchiffrer une documentation excessivement complexe afin d'aboutir au graphique voulu et pas à celui imposé par défaut. Autrement dit, il s'agit de savoir qui contrôle le résultat final, le statisticien ou le logiciel.

2 Environnement

2.1 Taille des graphiques

Le graphique demandé est tracé dans une zone dont les dimensions sont définies par les paramètres `hsize` et `vsize` de la commande globale `goptions` (les valeurs maximales sont prises par défaut) diminués de l'espace nécessaire à l'édition des titres, sous-titres, notes, légendes,.... Les dimensions peuvent être exprimées en trois unités : `pouce, cm` ou `pct` qui signifie "pourcentage de la dimension totale". Cette dernière unité est préférable pour exprimer les tailles de caractères et symboles lorsque les dimensions globales, liés au périphérique de sortie, sont sujettes à modifications.

2.2 Sauvegarde des graphiques

En l'absence de commande explicite, le graphe apparaît sur le périphérique par défaut, l'écran de l'ordinateur, dans la fenêtre de visualisation des résultats. Une fois les graphiques mis au point, ils peuvent être sauvés dans des fichiers ou un traitement de texte.

Un fichier de format `.jpg` est systématiquement créé (sous windows) dans un répertoire temporaire ainsi qu'un fichier `sashtml.html` contenant le graphique ; consulter la fenêtre du journal. Il devrait être possible de contrôler la destination de ce fichier.

2.3 Image écran

Un clic droit sur le graphique (windows mais unix ?) ouvre un menu qui permet de sauver l'image dans le format `.png` avec la définition de l'écran.

2.4 ODS

Comme vu en introduction, le graphique peut être automatiquement orienté dans une fichier au bon format (.rtf, .html...) et sans doute avec une meilleure définition que celle de l'écran.

```
ODS RTF BODY='nomfichier.rtf';
ODS GRAPHICS ON;
/* Programme SAS */
ODS GRAPHICS OFF;
ODS RTF CLOSE;
```

3 Commandes globales

Elles définissent des objets (axes, symboles, trames, légendes) et les options utilisés pour les tracés ; elles demeurent valables jusqu'à une nouvelle définition ou la fin de la session sas.

3.1 Axes

Des types d'axes, numérotés de 1 à 99 sont définis avant de pouvoir être utilisés dans les différents graphiques. Ils précisent l'échelle (liste de valeurs, logarithmique), l'apparence (longueur, couleur, épaisseur, style de ligne, origine), les marques d'échelle (nombre, couleur, épaisseur, hauteur), les valeurs des échelles (format), le libellé (police,...).

```
axis1 order=(1973 to 1981 by 2)
   label=('annee')
   minor=(number=1)
   width=3;
axis2 order=(0 to 10000 by 1000)
   label=('Revenu en francs')
   minor=none
   width=3;
```

3.2 Légendes

Comme pour les axes, différents types de légendes (de 1 à 99) sont définissables. Ils spécifient positions et textes des libellés qui identifient les différents graphismes et symboles utilisés.

3.3 Symboles

Les différents types de symboles (1 à 99) sont définis afin de décrire les modes de représentation recherchés. Sont concernés : le symbole (forme, taille, couleur) utilisé pour représenter un point, le type de lignes reliant les points (couleur, continue, hachurée, pointillée,...), la façon ou mode d'interpolation incluant barres, boîtes à moustaches, escaliers, splines, intervalles de confiance, régression (linéaire, polynomiale, spline).

```
symbol1 interpol=sm50s  /* lissage spline */
    value=diamond       /* symbole        */
    height=3            /* taille du symb.*/
    width=2;            /* epaisseur      */
```

3.4 Options graphiques

Outre ceux décrits ci-dessus (`hsize, vsize`), cette commande redéfinit les valeurs de plus de 80 paramètres affectant

- les différents aspects du graphique :
  - *border* cadre autour du graphique,

  - *gunit=cm|in|pct* unité de mesure,
  - *rotate=landscape|portrait* orientation du graphique,
- le texte :
  - *ftext* police du texte,
  - *ftitle* police des titres,
- texte, symboles, types de hachures, légendes.
- Les paramètres reprennent leurs valeurs par défaut à la suite de :
  - *reset=all|global all* concerne tous les paramètres tandis que que *global* n'affecte pas ceux définis dans la même commande.

3.5 Titres et notes

Les commandes `title` et `footnote` définissent des lignes de texte autour du graphique, elles suivent le même principe que celui décrit au paragraphe I.1.4 et d'autres options sont disponibles : taille, couleur et police des caractères, position, rotations de la ligne de texte et des caractéres, tracés de lignes.

```
goptions reset=global gunit=pct border
    ftext=swissb htext=3;
title1 height=5 'Institut';
title2          'de';
title3 height=5 'Mathématiques';
footnote1 font=script justify=left
    'Universite de Toulouse';
```

Il est important de noter que chaque paramètre peut être initialisé ou redéfini à différents endroits d'un programme SAS : dans les commandes spécifiques (`symbol, legend, axes, pattern, title, footnote`), par la commande `goptions` et dans chacune des procédures. Ceci impose de bien distinguer les paramètres globaux, applicables à tous les graphes, des paramètres spécifiques à chaque graphe.

4 procédure "gchart"

Cette procédure trace des diagrammes en barres (`hbar`), en colonnes et histogrammes (`vbar`), en secteurs (`pie`) et aréolaires (`star`). Elle peut traiter des variables quantitatives ou qualitatives ; les variables quantitatives sont codées explicitement ou automatiquement en classes ou, selon les besoins, sommées ou moyennées.

4.1 Syntaxe

proc gchart <options générales> ;
by <descending> variable ;
vbar liste de variables
</<options d'apparence>
<options statistiques> <options d'axes> > > ;
hbar liste de variables
</<options d'apparence>
<options statistiques>
<options d'axes> > ;
pie liste de variables
</<options d'apparence>
<options statistiques> ;
star liste de variables
</<options d'apparence>
<options statistiques> > ;

4.2 Options générales

- *data=table sas* indique le nom de la table ou, par défaut, la dernière crée,
- *annotate=* table contenant les compléments graphiques.

4.3 Options d'apparence

Elles spécifient les couleurs, les espacements et largeurs de colonnes ou barres. Il est également possible d'adjoindre un cadre (`frame`), de supprimer (`nolegend`) ou modifier la légende. Une option `annotate` peut être introduite au niveau de chaque commande.

4.4 Options statistiques

- *sumvar=* variable quantitative dont le cumul ou la moyenne est représenté,
- *freq=* variable de pondération des observations,
- *midpoints=* liste des bornes de classes,

- *levels*= nombre de classes,
- *type*= spécifie ce que représente le graphique (par défaut une fréquence) : `cfreq` (fréquence cumulée), `cpt` (pourcentage cumulé), `pct` (pourcentage), `sum` ou `mean` (associées à `sumvar=`).
- *group*= représentation de plusieurs graphes côte à côte suivant les modalités de la variable spécifiée (`hbar` ou `vbar`),
- *subgroup*= découpage des barres ou colonnes selon la participation des modalités de la variable spécifiée (`hbar` ou `vbar`).

4.5 Options d'axes

Deux options permettent de de définir les axes ou de leur assigner des déclarations antérieurs : `gaxis=axis`n pour l'axe des groupes et `maxis=axis`n pour celui des bornes où n caractérise la définition d'axe concernée (cf. paragraphe V.2.1.).

5 procédure "gplot"

Graphiques en haute résolution de nuages de points en deux dimensions.

5.1 Syntaxe

proc gplot <options générales> ;
by <descending> variable ;
plot liste de graphiques
</ < *annotate*=data-set >
< options d'apparence>
< options d'axes>> ;
bubble liste de graphiques
</ < *annotate*=data-set >
< options d'apparence>
< options d'axes>> ;

5.2 Options générales

- *data=table sas* indique le nom de la table ou, par défaut, la dernière crée,
- *annotate=table sas* table contenant les compléments graphiques.
- *uniform* impose les mêmes échelles aux axes des différents graphiques.

5.3 Options d'apparence

Elles spécifient les couleurs, les polices de caractères, les tailles des bulles (*bubble*), le hachurage d'aires, la définition de légendes, la superposition (`plot`).

5.4 Options d'axes

Deux options permettent de définir les axes ou de leur assigner des déclarations antérieurs : `vaxis=axis`n pour l'axe vertical et `haxis=axis`n pour l'axe horizontal où n caractérise la définition d'axe concernée (cf. paragraphe 5.2.1.). De plus, `frame` trace un cadre tandis que `noaxis` supprime les axes.

5.5 Commandes

by suivi du nom d'une variable qualitative indique que les graphiques sont tracés par groupe d'observations ; la table doit être triée.

plot liste des graphes sous la forme : `y*x<=`n|`variable`>, avec la même syntaxe que précédemment pour désigner plusieurs graphes (a*(a b),...). La variable y fournit les ordonnées et x les abscisses des points représentés par des symboles définis dans la commande `symbol`n ou par différents symboles selon les valeurs de la `variable` spécifiée qui induit une classification. Dans ce dernier cas, une légende est créée par défaut.

bubble liste des graphes sous la forme : `y*x = size` où `size` est une variable indiquant la taille des bulles à tracer autour des centres de coordonnées x et y.

6 Annotate data set

Une *table d'annotations*, définie lors d'une étape data, est une table SAS contenant les descriptifs d'un ensembles de graphiques qui viendront se superposer aux résultats des procédures précédemment décrites (`gchart, gplot, ...`). Il est alors possible de positionner tout libellé ou toute figure géométrique simple et ainsi de personnaliser ses graphiques.

Des applications immédiates sont, par exemple, la production de plans factoriels avec identifications des points (variables, individus, modalités) par des libellés explicites ou encore le tracé du cercle des corrélations en analyse en composantes principales.

Par principe, chaque ligne ou “observation” d’une table d’annotations est une commande de réalisation d’un graphique particulier. Les valeurs de chacune des “variables” spécifient comment réaliser ce graphique : type, emplacement, couleur,.... Les variables de la table d’annotation ont des noms prédéfinis ; les plus usuelles sont :

function indique ce qu’il faut tracé : *bar, draw, frame, pie, symbol, label,...*,

x positionnement en abscisses,

y positionnement en ordonnées,

size hauteur des caractères,

xsys unité de mesure des abscisses,

ysys unité de mesure des ordonnées,

hsys unité de mesure des hauteurs,

color couleur,

position d’un texte par rapport aux coordonnées (calé à gauche, centré,...),

line type de ligne (par défaut, continue),

text texte du libellé

style police de caractères.

La mise en œuvre de ces fonctionnalités est un peu fastidieuse mais c’est la seule façon de faire éditer par SAS certains types de graphiques dont les fameux plans factoriels avec les libellés en clair de tous les points.

Création d’une table d’annotations :

```
data annocomp;
  set outcomp;
  x    = prin1;
  y    = prin2;
  xsys= '2';
  ysys= '3';
  text= lib_ind;
  size= 0.8;
  label x = 'axe1';
  label y = 'axe2';
  keep x y text xsys ysys size;
run;
```

SAS macros : écriture de macros commandes

Résumé

Cette vignette décrit brièvement les principes et objets du macro langage de SAS permettant d'écrire des macros commandes : macros variables, macros fonctions, passages de paramètres et syntaxe d'une macro commande.

Retour au plan du cours.

1 Introduction

1.1 Motivations

Dès qu'il s'agit d'écrire des programmes SAS suffisamment généraux afin, par exemple, de les appliquer à différents jeux de données, il est nécessaire, par souci d'efficacité, de faire appel aux ressources de SAS permettant de définir des *macro-variables* et des *macro-commandes*.

L'écriture de macros SAS est une activité courante dans les grandes entreprises et les sociétés de service spécialisées dans ce logiciel. L'objectif est de concevoir puis carrosser un ensemble de traitement spécifiques afin de les rendre accessibles à des utilisateurs non spécialistes de SAS mais gros consommateurs comme dans l'industrie pharmaceutique ou le marketing.

1.2 Principes

Le principe général consiste à associer une *chaîne de caractères*, une suite de commandes ou, un texte à un *identificateur*. Par la suite, toute occurrence de cet identificateur ou macro variable est remplacée par le texte désigné au cours d'un traitement préalable à l'exécution proprement dite des commandes.

Le pré-processeur implicitement invoqué reconnaît différents objets : variables, commentaires, commandes, fonctions, arguments, qui lui sont propres (précédés des caractères `&` ou `%`) ; ils lui confèrent les possibilités d'un langage de programmation rudimentaire mais structuré.

Le macro-langage, au même titre, augmente les possibilités du langage de base. Il permet de passer des paramètres entre les étapes DATA et PROC et de systématiser l'enchaînement d'une séquence donnée d'instructions.

Les macro-variables et macro-commandes sont connues, sauf déclaration explicite contraire (`%global, %local`), dans l'environnement dans lequel elles sont déclarées : globalement pour toute une session SAS ou localement à l'intérieur d'une macro.

2 Macro-variables

2.1 syntaxe

La déclaration d'une macro-variable consiste à associer par la commande `%let` une chaîne de caractères (jusqu'à 65534) à un identificateur (de 1 à 32 caractères) :

```
%let nomvar1=taille;
%let nomvar2=poids;
%let varlist=csp sexe age taille poids revenu;
/* affichage du contenu : */
%put &varlist;
```

Certaines déclarations de macros variables sont implicites : compteur d'une boucle `%do`, paramètres d'une macro-commande,....

Dans la suite du programme, les références à une macro-variable sont précédées du caractère `&` :

```
title "Étude des variables &varlist";
proc plot;
     plot &nomvar1*&nomvar2;
run;
```

Le caractère `&` est traduit au niveau du pré-traitement : remplacer la macro-variable qui suit par la chaîne de caractères avant de passer à l'exécution.

Attention aux chaînes de caractères. Si la chaîne est entre "... ", une macro variable contenue dans la chaîne est interprétée, remplacée par sa "valeur". Ce n'est pas le cas si la chaîne est entre '...'.

Une macro-variable peut contenir elle-même des commandes ou instructions SAS mais, dans ce cas, il est préférable de définir une macro-commande.

Le système gère des macros variables prédéfinies comme `sysdate` et `sysday` qui contiennent respectivement la date et le jour du début de la session SAS en cours. Exécuter et consulter la fenêtre du journal :

```
%put &sysdate;
%put &sysday;
%put &_automatic_;
```

La dernière commande liste toutes les macros variables du système. Ces macros variables peuvent être incluses dans un programme pour dater les sorties.

2.2 call symput

L'instruction `call symput` permet de créer une macro-variables en lui affectant les valeurs d'une variable d'une table SAS. Cette instruction s'utilise exclusivement dans une étape DATA.

```
CALL SYMPUT("nom-macro-var",valeur-macro-var);
```

Cela signifie que le contenu de cette macro variable n'est pas prédéfini par une chaîne de caractères mais peut évoluer en cours d'exécution. Ainsi dans l'exemple :

```
data _NULL_;
   set table-sas;
   call symput("nobs",_N_);
run;
```

Comme la variable prédéfinie `_N_` est incrémenté par le numéro de l'observation courante, à l'issue de l'excécution la macro variable contient comme "valeur" le nombre d'observations de la `table-sas`. C'est évidemment très pratique quand cette information n'est pas connue a priori sur l'ensemble des tables qui seront traitées.

3 Macro fonction

Un ensemble de macros fonctions s'appliquant à des macros variables sont prédéfinies. En voici des exemples

```
%let semaine = lundi - mardi - mercredi - jeudi -
 vendredi - samedi - dimanche;
%let longueur = %length(&semaine);
%let jour2 = %scan(&semaine,2,'-');
```

Dans cet exemple, la macro-variable longueur contient la longueur de la chaîne de caractères semaine soit 63. La macro-variable jour2 correspond ensuite au 2ème mot de la chaîne (soit mardi). Le troisième argument indique que le séparateur est un tiret.

Par définition, une macro-variable sert à stocker du texte. Par exemple, la valeur d'une macro-variable à laquelle on affecte `1+2` est la chaîne de caractères `1+2` et non l'entier 3. Toutefois, il est possible de forcer le compilateur macro à effectuer l'opération avec la macro fonction `%eval` qui opère des soustractions, multiplications et divisions seulement sur des entiers à partir d'expressions contenant des macro-variables. Si la valeur évaluée contient des décimales, la valeur est tronquée à la partie entière.

```
%let i = 22;
%let j = &i/7; /* j contient 22/7 */
%let k = %eval(&i/7); /* k contient 3 */
```

4 Macro-commandes

4.1 Syntaxe d'une macro commande

La déclaration d'une macro-commande (ou *macro* tout court) suit les mêmes principes,

```
%macro impdat;
%* ceci est un commentaire;
proc print;
run;
%mend impdat;
```

elle peut inclure des paramètres qui deviennent des macro-variables locales :

```
%macro plot(yvar,xvar);
proc plot;
  plot &yvar*&xvar;
run;
%mend plot;
```

L'exécution d'une macro est invoquée en faisant précéder son nom du caractère `%` :

```
%impdat;
%plot(taille,poids);
```

4.2 Valeurs et ordre des paramètres

Les paramètres peuvent avoir des valeurs par défaut :

```
%macro plot(yvar=taille,xvar=poids);
proc plot;
  plot &yvar*&xvar;
run;
%mend plot;
```

Il n'est plus nécessaire de leur assigner des valeurs. Attention à l'ordre des paramètres qui est significatif dans le cas de paramètres sans valeurs par défaut. Sinon, le nom du paramètre est obligatoire au moment de l'appel. Par principe, les premiers paramètres d'une macro commande sont ceux les plus utilisés qui n'ont pas de valeur par défaut.

5 Exemple

Exemple rudimentaire d'une macro commande pour faire de l'Analyse en composantes principales à partir de la procédure `princomp`.

```
%macro acp(ldataset, lident, llistev, red=);
%global dataset ident listev;
%let dataset=&ldataset;
%let ident=&lident;
%let listev=&llistev;
%* Acp de dataset ;
%* ident : variable contenant les;
%*      identificateurs des individus;
%* llistev : liste des variables (numeriques);
%* par defaut : reduites sinon red=cov;

%* options  edition;
options linesize=80 pagesize=60 number;
title "A.c.p. des donnees de &dataset";
footnote;

proc princomp data=donnees
          outstat=eltpr out=compr
          vardef=N &red;
    var &listev;
run;
%mend acp;

/* acp r'eduite par d'efaut */
%acp(crime,staten,murder--auto);
```

6 Bibliothèque de macros commandes

6.1 Dans un fichier

Une macro-commande doit être déclarée avant toute utilisation au cours d'un même session SAS. Lorsque certaines macro-commandes sont régulièrement utilisées, il est pratique de pouvoir les conserver les unes à la suite des autres dans un fichier muni de l'extension `.sas`. En les ajoutant dans un fichier appelé `macros.sas`, par exemple, il suffira ensuite dans n'importe quel programme SAS de faire l'appel :

```
%INCLUDE "macro.sas";
```

6.2 Dans un répertoire

Il est aussi possible de stocker les macros dans une *bibliothèque* de macros spécifique à un utilisateur ou à un groupe. Les règles suivantes sont à respecter :

1. Chaque déclaration de macro et une seule est enregistrée dans un fichier qui a pour nom l'identificateur de cette macro suivi de l'extension `.sas` : `impdat.sas`,
2. Tous ces fichiers sont regroupés dans un ou des répertoires qui constituent la ou les bibliothèques,
3. le ou les chemins d'accès sont spécifiés une fois pour toute dans le fichier de configuration SAS de l'utilisateur. Encore faut-il avoir identifié et localisé ce fichier et pouvoir y avoir accès !

En début de chaque session, SAS charge le répertoire de macros.

La Procédure SAS/SQL

Résumé

Cette vignette décrit l'usage de la procédure SQL qui permet l'interrogation de tables SAS à l'aide du langage de requête standard.

Retour au plan du cours.

1 Introduction

Le langage SQL (Structured Query Language) est un langage d'interrogation de bases de données standardisé commun à la plupart des logiciels de base de données. La procédure **sql** étudiée dans cette vignette en constitue une implémentation dans la version 9.3 de SAS. Cette procédure permet d'extraire, corriger et mettre à jour des données dans une table SAS, souvent plus rapidement que par une étape data.

Le terme de *table* désigne toujours une table SAS, correspondant à un stockage de données propre à ce logiciel. On utilise également deux nouveaux types d'objets : les *vues* et les *index*. Une vue est le stockage d'une interrogation ou ensemble de requêtes : elle contient la description ou définition d'une table virtuelle. Une vue est donc une interrogation à laquelle on donne un nom, pour son usage ultérieur dans une autre procédure SAS. Le principal intérêt de définir une vue est le gain d'espace mémoire. Un index est un système de pointeurs permettant dans certains cas d'accéder plus rapidement aux informations contenues dans une table SAS.

2 Syntaxe de la procédure

2.1 Commandes

proc sql <options> ;

alter table déclaration de modification ;

create table déclaration de création ;

delete décl-destruction ;

describe décl-description ;

drop décl-suppression ;

insert décl-insertion ;

reset <options> ;

select décl-sélection ;

update décl-mise à jour ;

validate décl-évaluation ;

Remarques :

- il est inutile de répéter l'instruction **proc sql** avant chaque déclaration, sauf si l'on exécute une étape data ou si l'on fait appel à une autre procédure entre deux commandes de `sql`.
- l'instruction `run` n'est pas nécessaire.

2.2 Options

- *inobs=n* restreint le nombre d'observations traitées (par exemple dans une clause `where`) sur une table fournie en entrée de la procédure.
- *outobs=n* restreint le nombre d'observations traitées (par exemple insérées) dans une table retournée par la procédure.
- *feedback* rappelle la définition des vues parentes lors de la description d'une vue (commande `describe`).
- *noprint* pas d'édition

3 Détail des commandes

3.1 La commande `create`

Elle permet de créer des tables, des vues, ou des index, à partir d'autres tables ou d'autres vues.

Création d'une table

Syntaxe

create table nom-table **as** query-expression ;

create table nom-table **like** nom-table ;

create table nom-table (def-col <, def-col>) ;

La première syntaxe est utilisée pour stocker les résultats d'une interrogation. C'est une façon de créer des tables temporaires. La deuxième syntaxe est utilisée pour créer une table ayant les mêmes noms de variables et mêmes attributs qu'une autre table. La troisième syntaxe est utilisée quand on veut créer une table dont les colonnes ne sont pas présentes dans des tables déjà existantes. Les syntaxes 2 et 3 créent des tables vides, qu'il faut ensuite remplir avec la commande `insert`. Création d'une table SAS permanente dans la librairie `sql`

```
libname sql 'sql';
proc sql;
create table sql.statlab like  sasuser.statlab2;
create table sql.statlab2 as
    select sexenf, gsenf, tenf_n,
     penf_n, tenf_10, penf_10
    from sasuser.statlab2
    where (consm_n='nonfum');
```

Création d'une vue

Syntaxe

create view nom-vue **as** query-exp <**order by**item <, item>> ;

Une vue étant une interrogation stockée et ne contenant pas de données, on ne peut utiliser les instructions suivantes quand on se réfère à une vue : `insert`, `delete`, `alter`, `update`.

Création d'une vue à partir d'une table

```
create view labv2 as
       select sexenf, gsenf, tenf_n, penf_n,
        tenf_10, penf_10
       from sql.statlab2
       where (sexenf='fille');
proc print data=labv2; run;
```

Création d'un index

Un index stocke à la fois les valeurs des colonnes d'une table, et un système de directions qui permet d'accéder aux lignes de cette table à partir des valeurs de l'index. L'utilisation de l'index lors d'interrogations ou autres instructions de la procédure est déterminée par le système. L'index est automatiquement mis à jour quand on modifie la table à laquelle il est associé. Il permet d'améliorer la performance de certaines commandes, par exemple la comparaison d'une colonne indexée à une valeur constante à l'aide de l'expression `where`.

Syntaxe

create <unique > **index** nom-index **on** nom-table ;

Le mot-clé `unique` garantit que chaque valeur de la colonne indexée est unique. Ceci peut être utile quand on manipule des variables telles que le numéro de sécurité sociale. Création de l'index simple `gse` associé au groupe sanguin

```
proc sql;
create index gse on sql.statlab2 (gsenf);
```

Création de l'index composite `consm` associé à deux variables

```
proc sql;
create index consm on sql.statlab2 (consm_n,consm_10);
```

3.2 La commande `alter`

Elle permet d'ajouter ou de supprimer des colonnes dans une table SAS, ou d'en modifier les attributs (longueur, label, format).

Syntaxe

alter table nom-table

< **add** def-col <, def-col >>

< **modify** def-col <, def-col >>

< **drop** nom-col <, nom-col >> ;

Modification d'une table existante

```
alter table sql.statlab2
      add gender char(6);
```

3.3 La commande `delete`

Elle permet de supprimer des lignes dans une table.

Syntaxe

delete from nom-table < **where** sql-exp > ;

Suppression des lignes d'une table

```
delete from sql.statlab2 where gsenf='A+';
```

3.4 La commande `describe`

Elle donne la définition d'une vue, et des vues parentes si l'option `feedback` est spécifiée.

Syntaxe

describe view nom-vue ;

Description d'une vue

```
describe view labv2;
```

3.5 La commande `drop`

Elle permet de détruire indifféremment une table ou une vue.

Syntaxe

drop table nom-table < , nom-table > ;

drop view nom-vue < , nom-vue > ;

3.6 La commande `insert`

Elle permet d'ajouter des lignes à une table.

Syntaxe

insert into nom-table < (nom-col < , nom-col > >)

values (value < , value >) ;

Il existe deux autres manières d'utiliser la commande `insert` (voir l'aide en ligne).

Insertion de lignes dans une table

```
insert into sql.statlab2
       values ('fille','AB',0,0,0,0,'d')
       values ('garcon','AB',10,10,10,10,'e');
```

3.7 La commande `select`

Elle permet de sélectionner des colonnes dans une table, et d'afficher les résultats dans la fenêtre `output`.

Syntaxe

select liste d'objets **from** liste < **where** sql-exp > ;

3.8 La commande `update`

Elle permet de modifier les valeurs de certaines observations pour des colonnes d'une table existante.

Syntaxe

update nom-table **set** nom-col=sql-exp < **where** sql-exp > ;

Modification d'une table

```
update sql.statlab2
       set gender=sexenf;
```

3.9 La commande `validate`

Elle permet d'évaluer la syntaxe d'une interrogation sans l'exécuter, et retourne un message dans la fenêtre `log`.

Syntaxe

validate query-exp ;

Cette commande est essentiellement utile dans des applications utilisant des macro-variables. `validate` retourne alors une valeur indiquant si l'interrogation est valide grâce à la macro-variable `SQLRC` (SQL Return Code).

4 Comparaison de la procédure `sql` avec une étape `data`

On veut connaître la taille moyenne à 10 ans et par sexe des enfants dont la mère consommait entre 10 et 20 cigarettes par jour au moment de leur naissance. Écrire le programme permettant de calculer ces quantités à l'aide des procédures `summary`, `sort` et `print` ; puis comparer avec le programme suivant utilisant la procédure `sql`.

4.1 Une utilisation de la `proc sql`

```
proc sql;
  select sexenf , mean(tenf_10) as tmoy
  from sasuser.statlab2
  where consm_n='10a20cig'
  group by sexenf
  order by tmoy;
```