AgroParisTech

Initiation au logiciel *SAS*(9)

pour Windows

N. Coqué

UFR de Mathématiques, Département MMIP

Table des matières

Introduction **3**

1 Présentation **3**
1.1 Les tables *SAS* . . . 3
1.2 Les programmes *SAS* . . . 4
1.3 L'environnement de *SAS* . . . 5
1.3.1 Les fenêtres . . . 5
1.3.2 Les menus . . . 6
1.3.3 Les bibliothèques . . . 7
1.4 Les étapes pour faire un programme *SAS* . . . 7
1.4.1 Règles d'écritures des instructions *SAS* . . . 7
1.4.2 Structure d'un programme *SAS* . . . 8

2 Le stockage de données **8**
2.1 L'instruction DATA . . . 8
2.2 L'instruction INPUT . . . 9
2.2.1 Le mode liste . . . 9
2.2.2 Le mode colonne . . . 10
2.2.3 Le mode formaté . . . 11
2.3 La création de données : CARDS . . . 12
2.4 La localisation d'un fichier de données : INFILE . . . 13
2.5 L'importation de données : IMPORT . . . 13
2.6 Les tables SAS permanentes : LIBNAME . . . 14
2.7 Le référencement d'un fichier : FILENAME . . . 14
2.8 La lecture d'une table : SET . . . 14
2.9 L'écriture d'un fichier de données : FILE et PUT . . . 15
2.10 L'exportation des données : EXPORT . . . 15
2.11 L'exportation des résultats . . . 16
2.12 La rédaction de rapport : l'ODS. . . . 16

3 Les manipulations de données **18**
3.1 Les opérateurs et les fonctions . . . 19
3.2 La transformation de tables . . . 21
3.2.1 Concaténation de tables : SET . . . 21
3.2.2 Fusion de tables : MERGE . . . 21
3.2.3 Mise à jour des tables : UPDATE . . . 23
3.3 Les instructions . . . 23

4 Les procédures **25**
4.1 Options générales . . . 25
4.1.1 Instruction DATA = . . . 25
4.1.2 Instruction TITLE . . . 25
4.1.3 Instruction FOOTNOTE . . . 26
4.1.4 Instruction VARIABLES . . . 26
4.1.5 Instruction BY . . . 26
4.1.6 Instruction FREQ . . . 27
4.1.7 Instruction WEIGHT . . . 27

4.2 Les procédures générales . . . 27
4.2.1 proc CONTENTS : édition du contenu de fichier . . . 27
4.2.2 proc PRINT : impression de la table . . . 27
4.2.3 proc SORT : tri de la table . . . 28
4.2.4 proc GPLOT : tracé de graphique . . . 28
4.2.5 proc GCHART : tracé d'histogramme . . . 30
4.2.6 proc DOCUMENT : manipulation des ODS . . . 31
4.3 Les procédures de statistiques descriptives . . . 34
4.3.1 proc CORR : calcul des corrélations . . . 34
4.3.2 proc FREQ : tableau croisé, fréquence . . . 34
4.3.3 proc MEANS : moyennes . . . 35
4.3.4 proc UNIVARIATE : fractiles . . . 37
4.4 La régression linéaire : proc REG . . . 38
4.5 Le modèle linéaire général : proc GLM . . . 41
4.6 L'analyse en composantes principales : proc PRINCOMP . . . 44

Références **48**

Table des figures

1 Exemple d'un tableau de données . . . 4
2 Fenêtres SAS . . . 5
3 Graphes de la proc REG avec l'ODS Graphics . . . 17
4 Graphe de la fonction y=sin(2x) . . . 29
5 Rapport obtenu par la procédure DOCUMENT . . . 33
6 Structure des résidus . . . 40
7 Projection des individus sur les axes 1 et 2 . . . 46
8 Graphes de la proc PRINCOMP avec l'ODS Graphics . . . 47

Introduction

Le logiciel *SAS* est de conception américaine : il est développé et commercialisée par la société SAS-Institute, située à Cary, en Caroline du nord. A l'origine, *SAS* (Statistical Analysis System) est un logiciel de statistique polyvalent, c'est-à-dire susceptible de traiter pratiquement tous les domaines de la statistique. Il est assez ancien (ses débuts remontent aux années 1960) et est constamment enrichi de nouvelles méthodes. Par conséquent, il est très volumineux et souvent redondant : le même problème statistique peut être traité par différents modules du logiciel (avec souvent des présentations différentes !).

Le langage de commande de *SAS* est un langage de programmation de 4ème génération. Aujourd'hui, il est devenu un véritable système de gestion de l'information plutôt qu'un simple logiciel de statistique. Il constitue un logiciel privilégié pour la gestion de grandes bases de données. Le logiciel *SAS* est très répandu et a acquis une situation dominante dans beaucoup de secteurs d'activités.

1 Présentation

Le système *SAS* est un ensemble de modules pour la gestion et le traitement statistique des données. Il a pour vocation de :

- collecter les informations provenant des différents systèmes opérationnels, quels que soient leur source de données ou leur format,
- présenter les résultats, entre autres de façon graphique, aussi clairement que possible,
- effectuer des traitements statistiques classiques, de modélisation, de prévision, de data mining.

Les versions 8 et 9 proposent des solutions : guided data analysis, market project, time series forecasting, module Insight....qui sont associées à une interface graphique. Elles permettent un traitement de l'information sans écrire une ligne de programme. Ces solutions sont limitées, ce qui rend incontournable l'usage du langage de programmation *SAS*.

1.1 Les tables *SAS*

Prenons un exemple : vous souhaitez étudier les caractéristiques physiques d'un ensemble de personnes. Vous pourriez enregistrer pour chacune d'elles son nom, son sexe, son âge, sa taille et son poids. Le tableau (figure 1) illustre ce fichier de données.

Valeur. Chacune des informations que vous avez enregistrée - sexe de Jean, poids d'Hélène, taille de Mélanie, etc... - est une valeur. La valeur d'une donnée est une simple mesure : la taille d'une personne, son poids, etc ...

Observation. Les informations concernant chaque personne - nom, âge, sexe, taille, poids - forment une observation. Chaque ligne du tableau constitue donc une observation. **Une observation est l'ensemble des valeurs concernant un même individu**.

Variable. Les valeurs contenues dans une colonne du tableau constituent une variable. **Une variable est un ensemble de valeurs concernant une même caractéristique**, telle que

le poids d'une personne, etc... Les variables *SAS* peuvent être de type numérique ou alphanumérique. *SAS* identifie les variables par leur nom, il est donc conseillé d'utiliser des noms de variables rappelant leur contenu : NOM, AGE, POIDS plutôt que V1, V2, V3.

NOM	SEXE	AGE	TAILLE	POIDS
Albert	M	64	175	77.5
Louis	M	8	132	35.5
Mélanie	F	34	158	51.2
Jean	M	15	144	46.2
Pierre	M	12	140	32.3
Laura	F	24	166	55.5
Hélène	F	38	165	54.6
Cathy	F	24	167	65.2
Claudia	F	34	163	56.4
David	M	26	185	78.3

FIGURE 1 – Exemple d'un tableau de données

Valeur manquante. Une valeur manquante représente une valeur de donnée manquante ou non disponible. Elle est représentée par un blanc ou un point, en fonction de la méthode de saisie et de lecture des données (cf. chapitre 2.2).

Table *SAS*. Après saisie (cf. chapitre 2.3) ou importation (cf. chapitres 2.4 et 2.5), les données sont gérées par *SAS* sous la forme d'une table *SAS*, qui outre les données, contient diverses informations : nom de la table, date de création, nom et type de variables... Elle peut également être le résultat d'une précédente étape DATA (cf. chapitre 2.1) du programme ou d'un autre programme, ou bien d'une procédure.

Cette table de données *SAS* est construite dans un format spécifique illisible en dehors de *SAS*.

Une table *SAS* est soit temporaire, soit permanente. Lorsqu'elle est temporaire, elle n'est conservée que le temps de la session en cours. Elle est stockée dans un bibliothèque temporaire WORK (cf. chapitre 1.3.3) de *SAS*. Lorsqu'elle est permanente, elle est stockée dans la bibliothèque que vous spécifiez (LIBNAME, cf. chapitre 2.6). On peut la réutiliser pour une nouvelle procédure ou des manipulations.

1.2 Les programmes *SAS*

Un programme *SAS* est un enchaînement d'étapes de gestion des données (ce sont les étapes DATA) et d'appels de procédures, qui décrivent les traitements à réaliser (ce sont les étapes PROC, comme PROCédure).

L'étape DATA constitue une étape essentielle dans le traitement des données. Elle permet de lire les données et de les transformer. Elle offre la possibilité de lire des données :

- directement saisies au clavier (SAS/ASSIST), ou incluses dans le programme *SAS* (CARDS),
- contenues dans un fichier texte provenant d'un éditeur, d'un autre logiciel de statistique, d'un tableur ou encore d'un gestionnaire de base de données.

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 de format quelconque. Chaque étape *SAS* est une suite d'instructions demandant au système *SAS* d'accomplir certaines tâches. Une instruction comporte un mot clé indiquant à *SAS* la nature du travail à effectuer (création de tables, lancement d'un calcul, impression des données...) et se finit par un point-virgule.

L'utilisation de ce logiciel demande donc de créer :

- un programme écrit en langage *SAS* (à partir de la fenêtre Editeur de *SAS*, cf. chapitre 1.3.1)
- un fichier externe contenant les données, si celles-ci ne sont pas incluses dans le programme.

1.3 L'environnement de *SAS*

1.3.1 Les fenêtres

Cinq fenêtres de travail apparaissent successivement et peuvent être déplacées (cf. figure2).

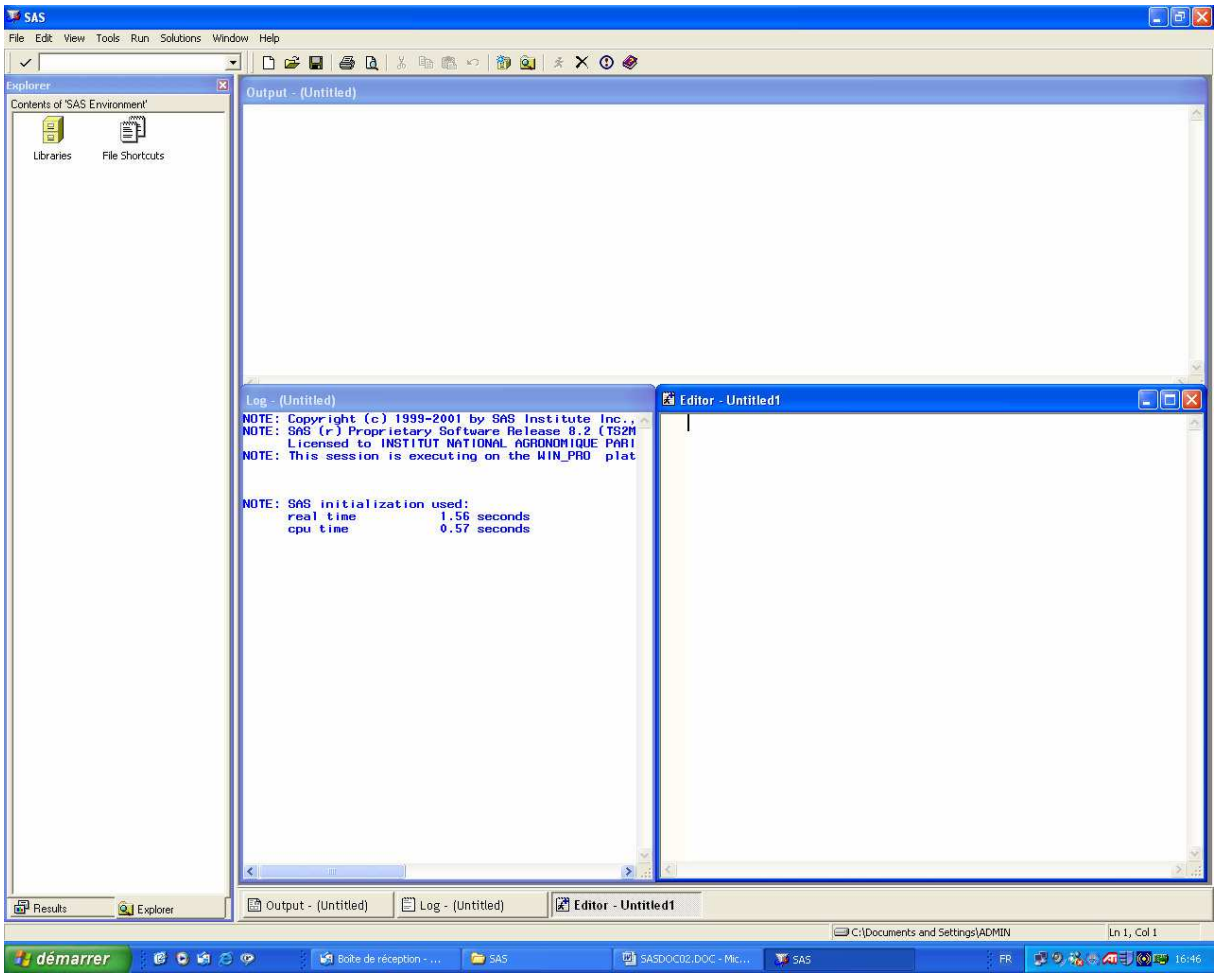

FIGURE 2 – Fenêtres SAS

Editor ou Editeur : la saisie du programme se fait dans cette fenêtre.
Deux éditeurs sont proposés : Program Editor et Enhanced Editor (éditeur amélioré), ce dernier est bien plus convivial car il aide, par la couleur, à écrire un programme *SAS*.

Output ou Sortie : les résultats sont édités dans cette fenêtre.
Cette fenêtre n'est pas nettoyée entre les exécution de programme, les résultats sont ajoutés les uns aux autres. Il est donc utile de la vider en utilisant la commande "Clear All" (soit par le menu déroulant, soit par un click droit de la souris) ou "Ctrl E".

Log ou Journal : le compte-rendu de l'exécution du programme apparaît dans cette fenêtre. Les commentaires en bleu traduisent le bon déroulement du programme, en vert, une anomalie et en rouge, une erreur. Cette fenêtre n'est également pas nettoyée entre les exécutions, il est donc souhaitable d'effacer régulièrement son contenu de la même façon que pour la fenêtre *Output*

Results ou Résultats : c'est une fenêtre de navigation dans les résultats.
Cette fenêtre permet de visualiser le menu des procédures utilisées dans le programme.

Explorer ou Explorateur : fenêtre de navigation dans les "bibliothèques" de données *SAS* (cf. chapitre 1.3.3).

Graph : dans le cas de programmes générant des graphiques (cf. chapitre 4.2.4), cette sixième fenêtre *SAS* est créée automatiquement.

1.3.2 Les menus

ToolBar : la barre d'outils permet de réaliser plus rapidement certaines commandes *SAS*. Des commandes peuvent être entrées directement depuis cette boîte qui peut être fermée ou réactivée par un click droit de souris. Quelques icônes intéressants :

- Break ou Tasking Manager : permet d'interrompre une exécution de programme
- Submit : permet d'exécuter un programme
- Help : permet d'accéder à l'aide en ligne *SAS*

Menu déroulant :

- File : gestion des fichiers (Open, Save, Import Data, Exit...)
- Edit : gestion de l'édition (Clear, Cut, Copy, Paste...)
- View : gestion des fenêtres, pour ouvrir une nouvelle fenêtre d'Edition de programme, ...
- Tools : pour utiliser des outils d'édition (de données, de textes, de graphiques, ...)
- Run : pour exécuter (Submit) tout ou partie d'un programme et rappeler l'exécution précédente
- Solutions : module de programmes clés en main pour utiliser des outils d'analyse, de développement, de mise en page, ...
- Windows : pour mettre en forme les fenêtres et sélectionner la fenêtre active
- Help : pour obtenir une aide en ligne (pour la version 8 : http ://v8doc.sas.com/sashtml/, pour la version 9 : directement par le menu)

Il existe également deux autres menus accessibles par un click gauche de souris sur l'icône à gauche du titre de la fenêtre, ou par un click droit dans la fenêtre.

1.3.3 Les bibliothèques

Le système *SAS* est composés de bibliothèques contenant des tables *SAS*. Ces bibliothèques permettent de stocker des tables *SAS* de manière permanente ou provisoire. Le stockage permanent permet de traiter les données sur plusieurs sessions. Les bibliothèques sont :

Work : bibliothèque **temporaire** n'existant que pendant la durée de la session *SAS*. Cette bibliothèque est utilisée par défaut. On y trouve toutes les tables de données générées par vos programmes. Ces tables peuvent être ouvertes par un double click. Elles doivent être refermées avant de relancer une procédure les utilisant !

SasUser : bibliothèque permanente associée à chaque utilisateur. Les tables de données sont accessibles par l'instruction :

```
data sasusuer.nomtab;
```

Il est particulièrement utile d'utiliser cette bibliothèque (ou une que l'on crée, voir chapitre 2.6) lorsque la table de données est de grande taille et qu'une mise en forme des données est faite dans un 1er temps (ex : normalisation des biopuces). Il suffit alors de rappeler ultérieurement cette table déjà prétraitée. Si la table n'est pas permanente, il faut réexécuter le prétraitement au début de chaque session.

SasHelp : bibliothèque contenant un groupe de catalogues qui permet le fonctionnement par défaut d'une session *SAS*, et surtout le système d'aide en ligne.

Maps : bibliothèque contenant des tables de données géographiques utilisées dans les exemples *SAS* (uniquement dans la version 8).

Nous verrons dans le chapitre 2.6 que l'on peut également créer ses propres bibliothèques.

1.4 Les étapes pour faire un programme *SAS*

1.4.1 Règles d'écritures des instructions *SAS*

- les instructions commencent par un mot clé (DATA, PROC, SET, etc...) et finissent par **un point virgule**.
- une instruction peut s'écrire sur plusieurs lignes et plusieurs instructions peuvent s'écrire sur une même ligne séparées par des points virgules. Cette dernière possibilité n'est pas recommandée pour des raisons de lisibilité.
- il est conseillé de décaler les instructions (indentation)
- pensez à insérer des commentaires dans vos programme : `/* commentaires */`
- ne jamais modifier directement les données de départ mais créer un tableau temporaire :

```
DATA donnee;
    SET tabtemp;    /* creation de la table tabtemp à partir de donnee */
RUN;
```

1.4.2 Structure d'un programme *SAS*

Étape 1 : Libname, localisation de la table. Cette étape est souvent omise, *SAS* cherche alors les tables dans la bibliothèque temporaire Work.

```
LIBNAME nombib "N:\Sas";
```

Étape 2 : DATA, création des tables *SAS*.

```
DATA nomtab;
```

Étape 3 : Proc, analyse des tables *SAS*. L'instruction PROC est utilisée pour appeler une procédure *SAS*. Les procédures sont des programmes qui lisent des tables, calculent des statistiques, éditent des résultats, créent de nouvelles tables. Leur forme générale est :

```
PROC nomproc DATA=nomtab;
```

Étape 4 : Submit, soumission de tout ou partie du programme.

Étape 5 : Log, vérification des erreurs dans la fenêtre Log.

Étape 6 : Lst, observation des résultats dans la fenêtre Lst.

Etape 7 : Export, exportation des résultats et des données vers Word ou Excel, par exemple. Cette étape est facultative.

2 Le stockage de données

Cette partie concerne la création ou la lecture de données. Les données peuvent ne pas avoir encore été encodées et vous pourrez alors choisir leur format, ou être déjà encodées et vous devrez connaître leur format. Les tables sont de forme rectangulaire avec les individus en lignes et les variables en colonnes.

NB : le séparateur des nombres décimaux est un point (.) et non une virgule (,), faites attention avec Excel !

2.1 L'instruction DATA

L'instruction DATA marque le début de l'étape DATA de création d'une table *SAS*. Elle est suivie du nom donné à la table :

```
DATA nomtab;
```

où *nomtab* est le nom de la table qui contiendra les données qui vont être introduites, lues ou manipulées (cf. exemple 1).

Cette table est un fichier de travail temporaire qui est créé par le logiciel dans la bibliothèque Work et qui est détruit à la fin du programme. Elle peut être permanente, et dans ce cas son nom est composé, la première partie du nom étant le nom de la bibliothèque (cf. chapitre 2.6). Cette table est nécessaire au programme et ne doit pas être confondue avec l'étape d'encodage des données.

2.2 L'instruction INPUT

Cette instruction permet de décrire les données au système *SAS*. Si les variables sont très nombreuses, une écriture condensée est possible : par exemple var1-var5 définira 5 variables codées respectivement var1 à var5. Si la variable est alphanumérique, le nom de la variable doit être suivi du caractère $ (cf. exemple 1).

Il y a trois principaux formats d'encodage des données pour *SAS* : liste, colonne et formaté.

2.2.1 Le mode liste

Dans ce cas, il faut séparer chaque valeur par un blanc et passer à la ligne à la fin de chaque enregistrement (cf. exemple 1).

Syntaxe de lecture :

```
INPUT  variable1 variable2 ...variablek;
```

où *variable1 variable2 ...variablek* est la liste des noms donnés aux variables, les noms étant séparés par des espaces.

Exemple 1

```
DATA  resultat;
INPUT  nom $ note1 note2 note3 classe $;
CARDS;
jean 44 25 33 6A
alexandre 34 33 34 6A
paul 22 44 21 6B
nicolas 28 36 40 6C
;
```

Ce format ne permet pas de lire :

- des variable alphanumériques de plus de 32 caractères ou contenant des blancs (puisque les champs blancs servent de séparateurs entre les données),
- les variables dans un ordre différent de celui de l'encodage .

Si le nombre de variables de l'instruction INPUT est :

- supérieur au nombre de valeurs présentes dans l'enregistrement, alors *SAS* utilisera l'enregistrement suivant pour fournir des valeurs aux variables. Il y aura de ce fait au moins deux enregistrements pour une observation. Cette erreur dans la lecture du fichier, fatale pour la validité du programme, ne sera pas détectée par le compilateur. C'est pour cela qu'il faut toujours contrôler dans la fenêtre LOG, que le nombre d'enregistrements lus corresponde au nombre d'enregistrements du fichier.
- inférieur au nombre de valeurs présentes dans l'enregistrement, alors *SAS* ignorera la fin de l'enregistrement. Vérifier par une procédure PROC PRINT ou une PROC CONTENTS que le nombre de variables lues corresponde au nombre de variables du fichier.

NB : une donnée manquante doit, obligatoirement, être codée par un point (.).

2.2.2 Le mode colonne

Un nombre constant de caractères est réservé pour chaque variable. Ceci nécessite que toutes les unités d'observation occupent le même nombre de colonnes (cf. exemple 2).

Syntaxe de lecture :

```
INPUT  variable1 debut-fin  variable2 debut-fin  variablek debut-fin;
```

où *début* et *fin* représentent les numéros de la première et de la dernière colonne de la variable concernée. Si la variable n'occupe qu'une colonne, un seul numéro est nécessaire.

Exemple 2

```
DATA  resultat;
INPUT  #1 nom $ 1-12 note1 #2 classe $ 8 note2 1-2;
CARDS;
de la boetie 35
34 33 6A
jean         44
25 33 6A
alexandre    34
33 34 6A
paul         22
44 21 6B
nicolas      28
36 40 6C
;
PROC PRINT;
RUN;
```

Obs	nom	note1	classe	note2
1	de la boetie	35	A	34
2	jean	44	A	25
3	alexandre	34	A	33
4	paul	22	B	44
5	nicolas	28	C	36

Remarques :

- *Une observation occupe deux enregistrements.*
- *La variable classe ne contient qu'un caractère (A, B ou C).*
- *La variable avant classe est ignorée.*
- *La variable classe est lue avant la variable note2.*
- *La variable note1 garde un format libre (séparée de nom par au moins un espace).*

Ce format permet de lire les variables dans n'importe quel ordre et d'ignorer volontairement certaines variables.

NB : une donnée manquante pourra être codée par un point (.) ou par un blanc ( ). Une variable de valeur nulle ne devra donc pas être codée par un ou plusieurs blancs car elle serait alors considérée comme une donnée manquante.

2.2.3 Le mode formaté

On peut associer des formats de lecture ("FORMATTED INPUT" dans la doc. *SAS*) aux variables. Nous citerons les formats les plus utilisés (pour des cas plus sophistiqués se reporter à la documentation *SAS*).

Lecture de données en continu (cf. exemple 3) : si la fin de chaque enregistrement n'est pas marquée par la touche Entrée, le format @@ permet de lire les données en continu (plusieurs observations sur la même ligne).

Lecture de dates (cf. exemple 4) : les formats dates permettent de lire et d'effectuer des opérations sur les dates. Une date est représentée par le nombre de jours depuis le 1 janvier 1960. Deux formats utiles :

- "DDMMYYw." permet de lire des dates sur w caractères (6 au minimum), sous la forme ddmmyy où dd est le jour du mois, mm le mois, et yy l'année.
- "MONYYw." permet de lire des dates sur w caractères (5 au minimum), sous la forme MMMyy où MMM sont les trois premières lettres du mois en anglais et yy l'année. La date exacte retenue pour les calculs correspond au premier jour du mois.

Exemple 3

```
DATA  pois;
INPUT rdt1 rdt2  @@;
CARDS;
56 45 43 46 60 50 45 48 66 57 50 50 65 61 60 63
60 58 56 60 53 53 48 55 60 61 50 53 62 68 67 60
;
PROC PRINT;
RUN;
```

Obs	rdt1	rdt2
1	56	45
2	43	46
3	60	50
4	45	48
5	66	57
6	50	50
7	65	61
8	60	63
9	60	58
10	56	60
11	53	53
12	48	55
13	60	61
14	50	53
15	62	68
16	67	60

Remarque : Le tableau pois contient 16 observations à 2 variables rdt1 et rdt2.

Exemple 4

```
DATA  dates;
INPUT jour1 DDMMYY8. jour2 MONYY6.;
CARDS;
13/09/06 oct06
01 08 06 sep06
;
PROC PRINT;
RUN;
```

Obs	jour1	jour2
1	17057	17075
2	17014	17045

2.3 La création de données : CARDS

Après avoir décrit la structure des données par l'instruction "INPUT", le début de l'entrée des données est signalé par l'instruction "CARDS ;" et la fin des données par un " ;" qui doit être placé sur une ligne après la dernière ligne de données (cf. exemple 5).

Les données sont ainsi intégrées au programme *SAS*. Ceci n'est intéressant que pour des données très peu nombreuses et associées à un programme particulier.

Exemple 5

Résultats d'un plan en bloc complet randomisé à un facteur qualitatif pour 2 variables x et y :

facteur	*variable*	*bloc 1*	*bloc 2*	*bloc 3*
niveau 1	*x*	*1.2*	*2.3*	*1.5*
	y	*137*	*165*	*143*
niveau 2	*x*	*1.0*	*2.0*	*1.1*
	y	*136*	*163*	*141*

```
DATA  plan;
INPUT bloc$ niveau$ x y;
CARDS;
1 1 1.2 137
1 2 1.0 136
2 1 2.3 165
2 2 2.0 163
3 1 1.5 143
3 2 1.1 141
;
```

Remarque :
Les deux premiers chiffres constituent l'identificateur (donc lus en alphanumériques) : le premier est le numéro du bloc, le second est le numéro du niveau du facteur.
On aurait pu également ne pas mettre le signe $ après bloc et niveau : les variables auraient alors été lues en tant que variables numériques.

2.4 La localisation d'un fichier de données : INFILE

Cette instruction indique au système *SAS* la localisation des données, et permet de lire des fichiers de données externes (fichier .xls, .csv, .txt, .don, etc...). Ce choix est à faire si le volume des données est important ou si les données doivent être utilisées par un autre programme. L'instruction est la suivante (cf. exemple 6) :

```
INFILE 'N:\ProjetStat\nomfic' options;
```

où :

- '*N :\ProjetStat *'est le chemin d'accès au fichier *nomfic*,
- *options* permet de rajouter :
  - le type de séparateur : *expandtabs* pour les tabulations (ex : fichier .txt), *dlm=';'* pour les fichiers .csv, etc...
  - la ligne à partir de laquelle la lecture commence : *firstobs=2* lorsque la 1ère ligne du fichier de données contient le nom des variables.

Exemple 6

```
DATA resultat;
    INFILE 'N:\ProjetStat\result.txt' expandtabs firstobs=2;
    INPUT nom$ note1 note2 note3 classe$;
RUN;
```

Remarque :

- *l'instruction DATA crée la table temporaire* resultat.
- *l'instruction INFILE lit le fichier de données* result.txt. *Les données sont séparées par des tabulations, la 1ère ligne n'est pas prise en compte.*
- *l'instruction INPUT (mode liste) lit les lignes de données brutes et donne le nom aux variables : ces noms peuvent être différents de ceux du fichiers d'origine* result.txt.

2.5 L'importation de données : IMPORT

Il est souvent utile d'importer un fichier de données externe (excel, par exemple). Il suffit alors d'utiliser la commande *Import Data* dans le menu *File* et de suivre pas à pas les indications.

La lecture des fichiers de données externes peut également se faire par la procédure IMPORT :

```
proc IMPORT datafile='N:\nomrep\nomfic' options;
```

Les options sont :

- *out=nomtab*, le nom de la table des données importées,
- *dbms=tab* si les données sont délimitées par des tabulations, *dbms=excel* pour les fichiers .xls, etc...(voir doc. *SAS*),
- *replace* permet de réécrire dans une table déjà existante.

2.6 Les tables SAS permanentes : LIBNAME

Si nécessaire, la table *SAS* peut devenir permanente (c'est-à-dire ne pas être détruite à la fin du programme) dans un répertoire déjà existant. Ceci n'est utile que si la table doit être utilisée dans d'autres sessions. Pour cela, il faut (cf. exemple 7) :

a) avant l'instruction DATA, définir une bibliothèque (le répertoire de stockage) :

```
LIBNAME nombib 'N:\ProjetStat\nomrep';
```

où *nombib* est le nom donné à la bibliothèque, et '*N :\ProjetStat*' est le chemin d'accès du répertoire *nomrep*.

b) associer le nom de la table, *nomtab*, à la bibliothèque par :

```
DATA nombib.nomtab;
```

Exemple 7

```
LIBNAME  note  'N:\ProjetStat';
DATA  note.essai;
INPUT note1 note2;
CARDS;
1 2
6 15
1.2 5.9
;
```

Remarque :

- *la bibliothèque* Note *apparaît dans la fenêtre* Explorer. *Cette nouvelle bibliothèque contient la table* Essai.
- *le fichier* essai.sas7bdat *est créé dans le répertoire 'N :\ProjetStat'. Cette extension n'est cependant pas indiquée dans le programme SAS. D'autre part, ce fichier est un fichier système SAS, il ne peut être ni édité ni visualisé, il n'est utilisable que par un programme SAS. Il pourra être rappelé ultérieurement (cf. exemple 8).*

2.7 Le référencement d'un fichier : FILENAME

De même que l'instruction LIBNAME sert à faire référence à un répertoire, l'instruction FILENAME sert à faire référence à un fichier. Sa syntaxe est :

```
FILENAME exemple "N:\ProjetStat\mesdonnees.txt";
DATA temporaire;
    INFILE exemple;
    INPUT...;
RUN;
```

2.8 La lecture d'une table : SET

Pour créer une table à partir de la lecture d'une autre table *SAS*, on utilise les instructions (cf. exemple 8) :

```
DATA nomtab1;
    SET nomtab2;    /*les valeurs de nomtab1 sont copiées dans nomtab2*/
RUN;
```

Lors de la lecture simultanée de plusieurs tables, une concaténation de tables est réalisée (cf. chapitre 3.2.1). Une table *SAS* peut être :

- temporaire, et dans ce cas son nom est simple,
- permanente, et dans ce cas son nom est composé, la première partie du nom étant le nom de la bibliothèque (cf. chapitre 2.6 et exemple 8).

Exemple 8

```
LIBNAME note 'N:\ProjetStat'; /*on associe la bib Note au répertoire*/
DATA new;                     /*création de la nouvelle table new   */
    SET note.essai;           /*dans la bibbliothèque temporaire    */
RUN;                          /*à partir de la table note.essai     */
```

2.9 L'écriture d'un fichier de données : FILE et PUT

Il est possible de conserver dans un fichier ASCII des données provenant d'une table *SAS*. Le fichier contenant les données est d'abord défini par l'instruction FILE, puis la structure des données est décrite en utilisant l'instruction PUT (cf. exemple 9).

L'écriture est équivalente à celles des instructions INFILE (cf. chapitre 2.4) et INPUT (cf. chapitre 2.2).

Exemple 9

```
DATA ecriture;
    INFILE 'N:\ProjetStat\result.txt';
    INPUT nom$ note1-note3 classe$;
    moyenne=(note1+note2+note3)/3;  /*Création de la variable moyenne,*/
    FILE 'N:\ProjetStat\moy.txt';   /*Création d'un fichier moy.txt   */
    PUT nom$ moyenne classe$;       /*contenant nom, moyenne et classe*/
RUN;

Fichier result.txt :
    jean 44 25 33 6A
    alexandr 34 33 34 6A
    paul 22 44 21 6B
    nicolas 28 36 40 6C

Fichier moy.txt :
    jean 34 6A
    alexandr 33.666666667 6A
    paul 29 6B
    nicolas 34.666666667 6C
```

2.10 L'exportation des données : EXPORT

Pour exporter simplement des données, il suffit d'utiliser la commande *Export Data* du menu *File*, et de suivre pas à pas les instructions.

On peut également utiliser la procédure PROC EXPORT. Le choix des options est le même que pour la procédure IMPORT (cf. chapitre 2.5).

```
PROC EXPORT DATA=nomtab OUTFILE=nomfic options;
```

2.11 L'exportation des résultats

Sauvegarde des résultats. Lors de la rédaction d'un rapport, il est parfois utile de mettre les résultats obtenus sous *SAS*. Vous pouvez :

- copier/coller simplement sous Word,
- enregistrer la fenêtre *Output* (*Save As* dans le menu *File*) dans un format .lst ou .rtf que vous ouvrirez ensuite avec Word.

Sauvegarde des graphiques. Pour enregistrer un graphique, placez-vous dans la fenêtre *Graph*, puis sélectionnez *Export As Image* dans le menu *File*, vous choisissez alors votre format (.gif, .jpeg, .ps, etc...). Cette fonction est également accessible par un click droit de souris dans la fenêtre *Graph*. Vous insérerez ensuite ce graphique dans votre rapport.

2.12 La rédaction de rapport : l'ODS.

Depuis la version 8, *SAS* propose un moyen rapide de disposer de rapports avec un formatage personnalisé au moyen de l'Output Delivery System (ODS). Les sorties de chaque procédure sont des objets et l'ODS est l'interface qui va mettre en forme cet objet.

Pour rediriger les sorties de certains blocs de votre programme vers un fichier externe, il suffit d'utiliser l'instruction ODS :

```
ODS RTF BODY='N:\ProjetStat\nomdoc.rtf';    /*fichier de sortie nomdoc*/
    /* Programme SAS */
ODS RTF CLOSE;
```

Vous pouvez également utiliser les formats html, Post-Script et Acrobat Reader, en remplaçant RTF (format lisible par Word) ci-dessus respectivement par HTML, PS et PDF.

Depuis la version 9, *SAS* propose l'ODS Graphics (ceci reste expérimental) qui ajoute à certaines procédures (PROC CORR, ANOVA, GLM, PRINCOMP, REG, etc...) un panel de 8 graphes (cf. exemple 10 et figure 3) que l'on peut directement inclure dans un rapport. Ces graphes sont beaucoup plus esthétiques que ceux obtenus par *Export As Image*.

Exemple 10

```
ODS RTF BODY='N:\ProjetStat\rapport.rtf';
ODS GRAPHICS ON;

PROC REG DATA = sashelp.class plots(unpack);
    /* plots(unpack) permet d'avoir des graphes individuels*/
    model Weight = Height;
RUN;
QUIT;

ODS GRAPHICS OFF;
ODS RTF CLOSE;
```

Remarque :

- *la table sas.help.class est composée de 19 individus et 5 variables : Name, Sex, Age, Height et Weight.*
- *on obtient le fichier rapport.rtf lisible par Word. Ce fichier contient les résultats et les graphes de la procédure Reg (régression, cf. chapitre 4.4).*

Le Système SAS

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

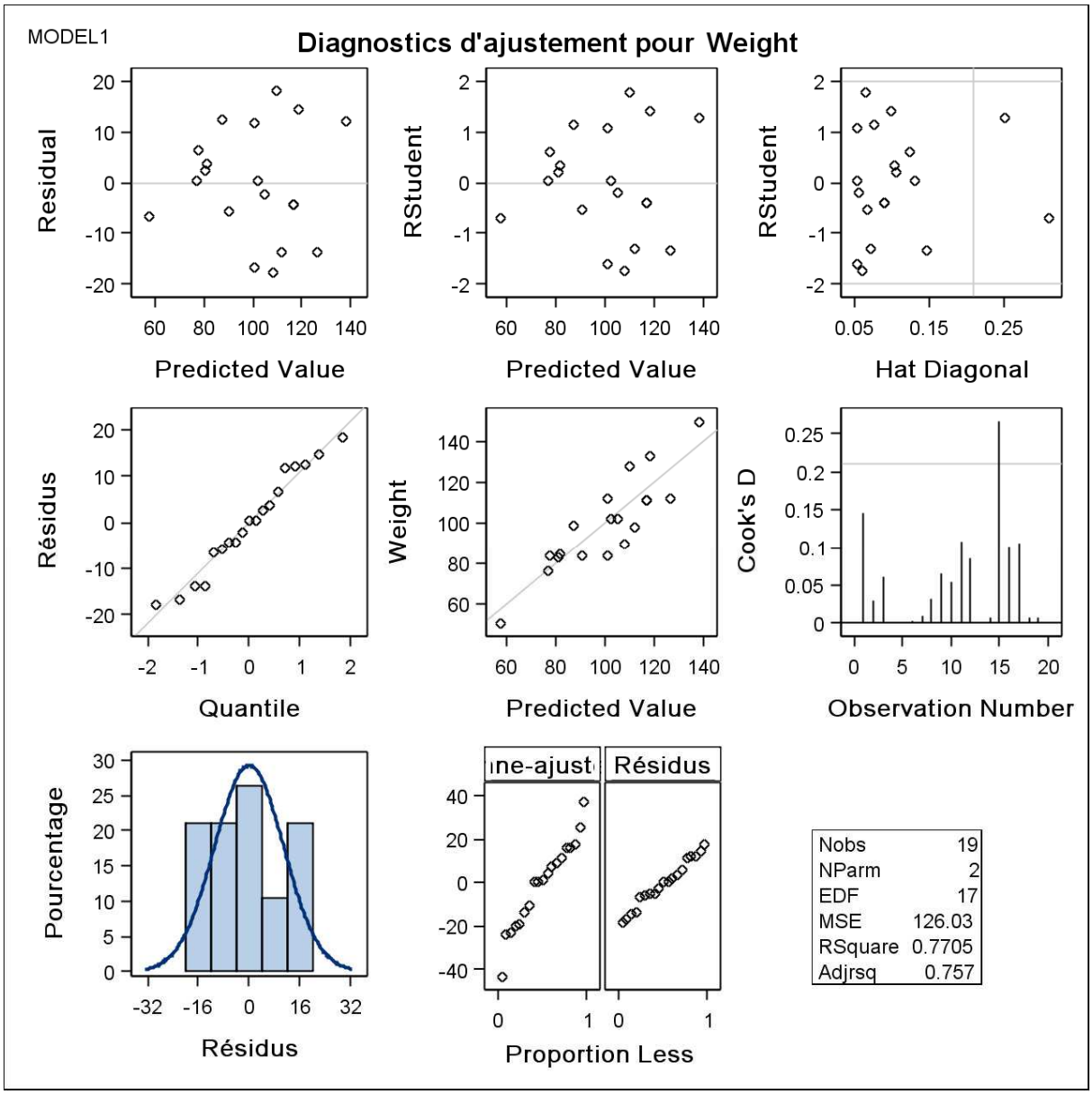

FIGURE 3 – Graphes de la proc REG avec l'ODS Graphics

Depuis la version 9, *SAS* propose également la procédure DOCUMENT (cf. chapitre 4.2.6) qui permet de manipuler les documents créés avec l'ODS. Elle permet de sélectionner les résultats et les graphes qui devront apparaître dans vos rapports, leur ordre d'affichage, etc...

3 Les manipulations de données

Des modifications des données (création de nouvelles variables, sélection d'observations) sont réalisables à partir d'instructions mais elles doivent être placées avant l'appel des procédures (cf. exemple 11, la première table *resultat*). Si ce n'est pas possible, il faudra créer une nouvelle table *SAS* avant de placer les instructions de manipulations (instruction SET utilisée dans la deuxième table *admis*).

Dans *SAS*, chaque instruction est effectuée pour chaque observation de la table DATA (ligne par ligne).

Exemple 11

```
DATA resultat;
INPUT nom$ note1 note2 note3;
somme=note1+note2+note3;            /*Création de la variable somme */
n=_n_;      /*n compte le nombre d'observations de la table résultat*/
IF somme>=100 AND note1>5
    THEN result='accepte';          /*Création de la variable result*/
    ELSE result='refuse';
CARDS;
jean 44 25 33
alexandre 34 33 34
paul 22 44 21
nicolas 28 36 40
;
    /*result ne prendra la valeur 'accepte' que si la variable somme*/
    /*est au moins égale à 100 et la variable note1 supérieure à 5  */
PROC PRINT;
RUN;
```

Obs	nom	note1	note2	note3	somme	n	result
1	jean	44	25	33	102	1	accepte
2	alexandr	34	33	34	101	2	accepte
3	paul	22	44	21	87	3	refuse
4	nicolas	28	36	40	104	4	accepte

```
DATA admis;
    SET resultat;
    IF result='accepte';
    moyenne=MEAN(note1,note2);
    /*La table admis ne contient que les observations telles que    */
    /*result='accepte'. On crée la variable moyenne qui est la     */
    /*moyenne des variables note1 et note2.                        */
PROC PRINT;
RUN;
```

Obs	nom	note1	note2	note3	somme	n	result	moyenne
1	jean	44	25	33	102	1	accepte	34.5
2	alexandr	34	33	34	101	2	accepte	33.5
3	nicolas	28	36	40	104	4	accepte	32.0

3.1 Les opérateurs et les fonctions

SAS possède des opérateurs et des fonctions permettant de créer de nouvelles variables ou de modifier les valeurs de variables existantes.

Les opérateurs arithmétiques habituels : +, -, *, / , **2 (au carré), **3 (au cube)

Les opérateurs logiques :

OR	|	ou, ou bien
AND	&	et, les deux
NOT		non, négation

Les opérateurs de comparaisons :

LT	<	inférieur à
GT	>	supérieur à
EQ	=	égal à
LE	<=	inférieur ou égal à
GE	>=	supérieur ou égal à
NE		différent de
NL		non inférieur à
NG		non supérieur à

Les fonctions de troncature :
SUBSTR extrait une partie de chaîne de caractère (cf. exemple 12) :

```
SUBSTR(argument, position, longueur)
```

ROUND arrondit les valeurs à l'unité la plus proche (cf. exemple 13) :

```
ROUND(argument, unité d'arrondi)
```

INT retourne la valeur entière (cf. exemple 13) :

```
INT(argument)
```

Exemple 12

```
DATA bureau;
INPUT lieu$;
batiment=SUBSTR(lieu,1,1);
piece=SUBSTR(lieu,2,3);
CARDS;
A108
C211
PROC PRINT;
RUN;
```

Obs	lieu	batiment	piece
1	A108	A	108
2	C211	C	211

Exemple 13

```
DATA arrondi;
INPUT x;
arrondi=ROUND(x,1);
dixieme=ROUND(x,.1);
centaine=ROUND(x,100);
manq=NMISS(x,arrondi);
somme=SUM(arrondi,dixieme);
entier=INT(x);
CARDS;
.
326.5436
1977.081
PROC PRINT;
RUN;
```

Obs	x	arrondi	dixieme	centaine	manq	somme	entier
1	.	.	.	.	2	.	.
2	326.54	327	326.5	300	0	653.5	326
3	1977.08	1977	1977.1	2000	0	3954.1	1977

Les fonctions de statistique descriptive. Elles sont de la forme :

```
fonction_descr(argument1, argument2,...argumentk)
```

ou bien :

```
fonction_descr(OF argument1-argumentk)
```

MAX	retourne le plus grand des arguments
MIN	retourne le plus petit des arguments
RANGE	retourne l'étendue des arguments
MEAN	retourne la moyenne des arguments
STD	retourne l'écart-type des arguments
VAR	retourne la variance des arguments
CV	retourne le coefficient de variation
SUM	retourne la somme des arguments
NMISS	retourne le nombre de valeurs manquantes
KURTOSIS	retourne le coefficient d'aplatissement
SKEWNESS	retourne le coefficient d'asymétrie
etc...	

Les fonctions mathématiques. Elles sont de la forme :

```
fonction_math(argument)
```

ABS	retourne la valeur absolue
ERF	retourne la valeur de la fonction d'erreur
EXP	retourne la valeur de la fonction exponentielle
GAMMA	retourne la valeur de la fonction gamma
LOG	retourne le log népérien
LOG10	retourne le log en base 10
LOG2	retourne le log en base 2
SQRT	retourne la racine carrée
etc...	

Les fonctions trigonométriques : SIN, COS, TAN, ARCOS, ARSIN, ATAN. Elles s'écrivent :

```
fonction_trigo(argument)
```

Les fonctions de probabilité : UNIFORM, NORMAL, POISSON, PROBBETA, PROBBNML, PROBCHI, PROBF, PROBT, etc...(voir la doc *SAS*)

3.2 La transformation de tables

3.2.1 Concaténation de tables : SET

La concaténation de tables consiste à réunir deux tables *SAS* contenant des observations différentes mais portant sur les mêmes variables : c'est donc un ajout de lignes. Cette instruction s'écrit :

```
SET  table1 table2...tablek;
```

où *table1 table2...tablek* est la liste des noms des tables *SAS*, séparés par des espaces.

Attention, les variables des tables doivent être les mêmes, sinon, le système *SAS* génère des données manquantes !

3.2.2 Fusion de tables : MERGE

La fusion de tables *SAS* est utilisée pour mettre en regard les observations de deux ou plusieurs tables *SAS* (cf exemple 14) :

```
MERGE table1 table2 ...tablek;
```

où *table1 table2 ...tablek* est la liste des noms des tables *SAS*, séparés par des espaces.

La table finale aura autant de lignes que la plus grande des tables, les valeurs des plus petites tables étant mises en données manquantes. Si une variable d'une table porte le même nom que celle d'un autre table, seule la valeur de la dernière variable est retenue.

On peut également fusionner des tables de façon contrôlée via des clés de fusion :

```
MERGE table1 table2;
BY variables;
```

Toutes les tables doivent contenir les variables citées dans l'instruction BY et être triées sur ces variables (cf chapitre 4.2.3). La fusion s'opère en joignant les observations correspondant aux mêmes valeurs des variables BY.

Exemple 14

```
DATA noms;
    INPUT nom$ classe$;
    CARDS;
jean 6A
alexandre 6A
nicolas 6C
paul 6B
;
PROC SORT DATA=noms; BY nom;    /*tri des tables 'noms' et 'admis' */
PROC SORT DATA=admis; BY nom;   /*(ex 11) sur la variable 'nom'    */
PROC PRINT DATA=noms;
PROC PRINT DATA=admis;
RUN;
```

Obs	nom	classe
1	alexandr	6A
2	jean	6A
3	nicolas	6C
4	paul	6B

Obs	nom	note1	note2	note3	somme	n	result	moyenne
1	alexandr	34	33	34	101	2	accepte	33.5
2	jean	44	25	33	102	1	accepte	34.5
3	nicolas	28	36	40	104	4	accepte	32.0

```
DATA fin;
    MERGE noms admis;           /*fusion des tables 'noms', 'admis'*/
    BY nom;
    KEEP nom classe note1;      /*on ne garde que ces variables    */
PROC PRINT;
RUN;
```

Obs	nom	classe	note1
1	alexandr	6A	34
2	jean	6A	44
3	nicolas	6C	28
4	paul	6B	.

Remarque : ne pas oublier de trier les tables sur la variable du BY (cf. chapitre 4.2.3).

3.2.3 Mise à jour des tables : UPDATE

UPDATE est un type spécial de fusion. Cette instruction est utilisée pour changer des valeurs de données d'une table *SAS* ou pour ajouter des observations à une table *SAS*.

Il s'agit d'une fusion particulière de deux tables, les observations doivent être triées par l'instruction BY. Les traitements sont effectués comme dans l'instruction MERGE...BY, mais l'instruction UPDATE ne conserve qu'une seule ligne correspondant au dernier traitement concernant cette modalité. L'instruction UPDATE ne retient que les valeurs non manquantes.

3.3 Les instructions

Les instructions de programmation (if, then, rename,..) permettent de créer, modifier, supprimer, renommer des variables et d'ajouter ou supprimer des observations (cf.exemple 14).

variable = expression : crée une nouvelle variable à partir d'autres variables (cf. exemple 10).

variable + expression : ajoute la valeur de *expression* à la valeur de *variable* à chaque passage.

IF condition ; : n'effectue l'action précédente que si la condition est vraie.

IF condition THEN instruction_1 ; : si la condition est vraie, l'instruction_1 sera exécutée, sinon elle sera ignorée, sauf si cette ligne est suivie de **ELSE instruction_2 ;** auquel cas l'instruction_2 sera exécutée.

IF condition THEN DO ; instruction_1 ;... ; instruction_k ; END ; : si la condition est vraie, les k instructions seront exécutées, sinon, elles seront ignorées.

KEEP variable_1 ... variable_k ; : garde dans la table en cours les variables citées. Il est recommandé de faire figurer cette instruction juste après l'instruction DATA.

DROP variable_1 ... variable_k ; : supprime de la table en cours les variables citées.

RENAME ancien-nom-de-variable_1 = nouveau-nom-de-variable_1...ancien-nom-de-variable_k = nouveau-nom-de-variable_k ; : cette instruction n'a d'effet que sur la structure de la table *SAS* en sortie. Par conséquent, dans l'étape de création de table, la variable doit être utilisée sous son "ancien nom". Dès que cette table sera créée, *SAS* ne reconnaîtra plus la variable que sous son nouveau nom.

OUTPUT : instruction permettant de contrôler l'ajout d'une observation dans le(s) table(s) en voie de création (cf. exemple 15).

Exemple 15

```
DATA hommes femmes;
INPUT nom$ poids taille age sexe$;
IF sexe='f' THEN DO;
    DROP sexe;
    OUTPUT femmes;
END;
ELSE DO;
    DROP sexe;
    OUTPUT hommes;
END;
CARDS;
claude 62 172 51 f
frederique 55 169 39 m
jacques 75 175 60 m
catherine 45 159 55 f
;
PROC PRINT DATA=hommes;
PROC PRINT DATA=femmes;
RUN;
```

Obs	nom	poids	taille	age
1	frederiq	55	169	39
2	jacques	75	175	60

Obs	nom	poids	taille	age
1	claude	62	172	51
2	catherin	45	159	55

4 Les procédures

Les procédures traitent les données définies à l'étape DATA et le plus souvent éditent des résultats.
Les procédures peuvent être :

- générales et se trouvent dans le manuel SAS/BASE,
- statistiques et se trouvent dans le manuel SAS/STAT,
- graphiques et se trouvent dans SAS/GRAPH,
- appliquées à des séries chronologiques et sont dans SAS/ETS,
- des techniques de recherche opérationnelle : SAS/OR (non disponible à AgroParisTech),
- des techniques de contrôle de qualité : SAS/QC (non disponible à AgroParisTech),
- mais également : SAS/ACCESS, SAS/AF, SAS/ASSIST, SAS/CONNECT, SAS/EIS, SAS/FSP, SAS/GIS, SAS/INSIGHT, etc...

L'appel d'une procédure se fait, après une instruction DATA et une fois toutes les manipulations de données et de variables faites, par :

```
PROC nomproc;
```

Des options peuvent préciser l'instruction PROC. Elles dépendent le plus souvent de la procédure utilisée (voir la doc. *SAS*), cependant, il en existe d'assez générales (cf. chapitre 4.1).
Par défaut, une procédure travaille sur :

- la dernière table créée,
- l'ensemble des observations,
- la totalité des variables.

4.1 Options générales

(cf. exemple 16)

4.1.1 Instruction DATA =

On peut spécifier la table *SAS* sur laquelle la procédure s'applique par :

```
PROC nomproc DATA=nomtab;
```

où :

- *nomproc* est le nom de la procédure,
- *nomtab* est le nom de la table *SAS*.

Par défaut, une procédure est appliquée sur la dernière table créée.

4.1.2 Instruction TITLE

On peut associer un titre à une procédure après l'instruction PROC, par :

```
TITLE nomtexte;
```

où *nomtexte* est le texte du titre.
Le titre sera imprimé sur la page éditée par la procédure, mais aussi par toute procédure ultérieure jusqu'à la rencontre d'une autre instruction "TITLE;". Pour invalider tous les titres, on utilise l'instruction :

```
TITLE;
```

4.1.3 Instruction FOOTNOTE

Cette instruction définit le texte à imprimer en bas des pages de sortie :

```
FOOTNOTE note;
```

Pour l'invalider, on utilise l'instruction :

```
FOOTNOTE;
```

4.1.4 Instruction VARIABLES

L'instruction VAR permet de restreindre le nombre de variables numériques de la table aux variables citées :

```
VAR nomvar1 nomvar2 ...nomvark;
```

où *nomvar1 nomvar2 ...nomvark* est la liste des noms des variables, séparés par un espace.
Par défaut, la procédure est appliquée à toutes les variables numériques de la table.

4.1.5 Instruction BY

Cette instruction permet d'appliquer la procédure à chaque sous-groupe défini par les valeurs de la ou des variables spécifiées :

```
BY nomvar;              */tri croissant*/
BY DESCENDING nomvar;   */tri décroissant*/
```

NB : la table est préalablement triée par ordre croissant (ou décroissant) sur cette variable (PROC SORT cf. chapitre 4.2.3).

Exemple 16
Reprenons l'exemple 15 :

```
PROC SORT DATA=hommes OUT=candidats;    /* on trie la table candidats*/
    BY DESCENDING age;                  /* tri decroissant */

PROC PRINT DATA=candidats;
    TITLE 'liste des candidats';
    FOOTNOTE 'MAJ mai 2006';
    VAR nom age;
RUN;
```

```
        liste des candidats                12:10 Monday, June 12, 2006 1

       Obs      nom       age
        1     jacques       60
        2     frederiq      39

            MAJ mai 2006
```

4.1.6 Instruction FREQ

Cette instruction spécifie une variable numérique dont la valeur représente la fréquence de l'observation :

```
FREQ nomvar;
```

Si cette valeur est inférieure à 1 ou si elle est manquante, alors la procédure n'utilise pas cette observation pour le calcul des statistiques.

4.1.7 Instruction WEIGHT

Cette instruction spécifie une variable numérique dont la valeur représente le poids de la variable analysée :

```
WEIGHT nomvar;
```

Si le poids est égal à 0, alors la procédure prend en compte l'observation dans le nombre total des observations.

Si le poids est inférieur à 0, alors la procédure converti cette valeur en 0 et compte l'observation dans le nombre total.

Si le poids est une valeur manquante, alors la procédure exclut l'observation de l'analyse.

4.2 Les procédures générales

4.2.1 proc CONTENTS : édition du contenu de fichier

```
PROC CONTENTS DATA=nomlib.nomfichier;
```

Cette procédure produit dans la fenêtre *Output* un résumé des caractéristiques de la table *nomlib.nomfichier* : information sur le contenu, variables, labels, formats, etc...

```
PROC CONTENTS DATA=WORK.mesdonnees;
RUN;
```

Cette procédure est également utilisée pour lister le contenu de toute une bibliothèque :

```
PROC CONTENTS DATA=WORK._ALL_;
RUN;
```

4.2.2 proc PRINT : impression de la table

```
PROC PRINT DATA=nomtab;
    BY variables;
    VAR variables;
RUN;
```

Elle édite les observations de la table *nomtab*. Par défaut, toutes les variables de la table sont listées. Comme pour toutes les procédures, l'instruction VAR permet de sélectionner les variables utilisées, et l'instruction BY de définir des sous-groupes.

4.2.3 proc SORT : tri de la table

Cette procédure permet de réordonner une table en fonction d'une ou plusieurs variables de tri (en ordre croissant par défaut). Cette procédure est très importante car beaucoup nécessitent en entrée des tables triées (par exemple, à chaque fois que l'on utilise un BY).

```
PROC SORT option1;
BY option2 var1 option2 var2 ...;
```

trie par ordre croissant les données selon les valeurs des variables de l'instruction BY sachant que la première variable sera la première clé de tri (cf. exemple 16).

option1 : pour conserver parallèlement la table non triée, il faut créer un fichier de sortie :

```
PROC SORT OUT=nomtab;
```

où *nomtab* est la table qui contiendra les données triées.

option2 : l'option DESCENDING placée devant le nom de la variable provoque un tri par ordre décroissant sur cette variable.

NB : cette procédure est la seule à ne pas éditer de résultats. Pour visualiser le contenu de la table triée, il faut utiliser la procédure d'impression :

```
PROC PRINT;
```

4.2.4 proc GPLOT : tracé de graphique

Cette procédure permet de tracer un graphique à deux dimensions (cf. exemple 17). Cette procédure graphique finit par l'instruction "QUIT ;".

Syntaxe :

```
SYMBOLn V=valeur I=méthode C=color;
PROC GPLOT DATA=nomtab;
    PLOT ordonnée*abscisse /options;
    BY variables;
RUN;
QUIT;
```

Représentation des points. Par défaut, chaque point est représenté par une croix, mais on peut représenter :

- chaque point par un caractère :

```
PLOT ordonnee*abscisse='caractère';
```

- des groupes de points selon une $3^{\text{ème}}$ variable :

```
PLOT ordonnée*abscisse=3emevariable;
```

- plusieurs graphiques consécutifs :

```
PLOT ordonnee1*abscisse1 ordonnee2*abscisse2...ordonnek*abscissek;
```

Les options. Elles sont séparées par un espace :

- OVERLAY permet de placer plusieurs graphiques sur un même dessin.
- HAXIS=*valeur* et VAXIS=*valeur* contrôlent les bornes et les graduations de l'axe horizontal et vertical. Valeur est la liste des valeurs désirées :
  – 10 TO 100 BY 10 : graduation de 10 en 10, de 10 à 100,
  – 10 100 1000 : 3 graduations.
- HREF=*valeur* et VREF=*valeur* permettent de tracer une ligne verticale à partir de la valeur HREF sur l'axe horizontal et une ligne horizontale à partir de la valeur de VREF sur l'axe vertical.

Exemple 17

```
DATA sinus;
    DO x=0 TO 5 BY .05 ;        /*de 0 à 5 tous les 0.05          */
    y=x*sin(2*x);
    OUTPUT;                     /*ajout d'observations dans 'sinus'*/
END;
PROC GPLOT;
    PLOT y*x='+' / VREF=0 ;     /*ligne horizontale en y=0        */
    TITLE y=sin(2x);            /*titre du graphique              */
RUN;
QUIT;
```

On obtient alors le graphique 4.

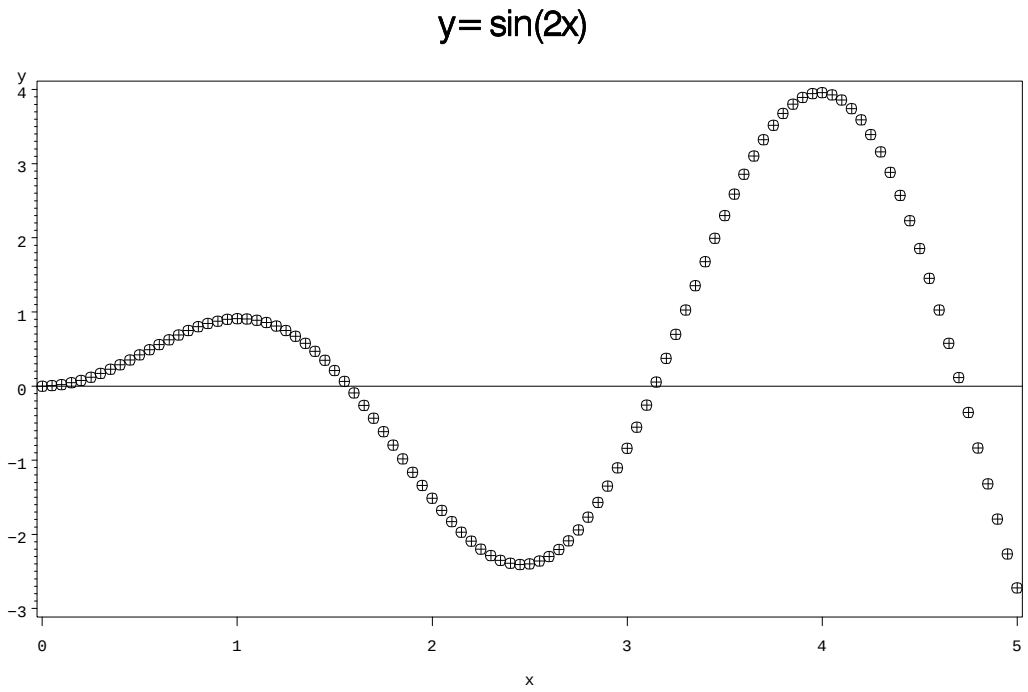

FIGURE 4 – Graphe de la fonction y=sin(2x)

La gestion des symboles. L'instruction SYMBOL placée avant PROC GPLOT, gère la représentation des points :

```
SYMBOLn V=valeur I=méthode C=color;
```

où :

- n est le numéro permettant de gérer plusieurs symboles à la fois,
- *valeur* est le code du symbole qui représentera chaque point sur le graphique. Par exemple :
  - *dot* pour un point,
  - *plus* pour le signe +,
  - *x* pour une croix,
  - *star* pour une étoile,
  - *square* pour un carré,
  - *triangle*, *diamond*, *circle*, etc...
- *méthode* est la méthode d'interpolation entre les points. Les plus habituelles sont :
  - *I=join*, joint les points entre-eux dans l'ordre d'apparition de leurs valeurs dans la table, une procédure de tri est parfois nécessaire,
  - *I=rlcli*, effectue une régression linéaire avec tracé des intervalles de confiance pour les valeurs,
  - *I=rq*, effectue une régression quadratique,
  - *I=spline*, effectue un lissage par spline.
- *color* est la couleur des symboles (back, blue, green, red...). Elle est à préciser même si les couleurs sont identiques pour chaque symbole.

Remarque :

1. Si l'instruction SYMBOL est unique, tous les points utiliseront le même symbole.
2. L'instruction SYMBOL reste active pendant toute la session *SAS*. Pour annuler la méthode d'interpolation, on écrit : "I=none".

4.2.5 proc GCHART : tracé d'histogramme

La procédure GCHART produit des histogrammes et calcule les fréquences (cumulées ou non), les pourcentages (cumulés ou non), les sommes et les moyennes.

Syntaxe :

```
Syntaxe :
PROC GCHART;
  BY variables;
  HBAR variables / options;
  VBAR variables / options;
  BLOCK variables / options;
  PIE variables / options;
  STAR variables / options;
RUN;
QUIT;
```

Les instructions :

- HBAR et HBAR3D éditent des histogrammes en bâtons horizontaux.
- VBAR et VBAR3D éditent des histogrammes en bâtons verticaux.
- BLOCK édite des histogrammes en blocs.
- PIE et PIE3D éditent des camemberts.
- STAR édite des graphiques en étoiles.

Les options :

- DISCRETE spécifie que les valeurs sont de type discret (nombre limité de valeurs numériques).
- MIDPOINTS=*valeurs*, valeur est le point moyen de chaque "barre" ou "section" de l'histogramme.
  - MIDPOINTS=10 TO 100 BY 10, définit un histogramme à 10 barres, centrées sur les valeurs 10, 20 ...100.
  - MIDPOINTS='bleu' 'vert' 'rouge', définit un histogramme à 3 barres pour ces valeurs discrètes.
- GROUP=*variable*, où variable est le nom de la variable discrète qui définit des groupes. Les histogrammes de chaque groupe sont alors construits côte à côte.
- SUBGROUP=*variable*, où variable est le nom de la variable qui définit des sous-groupes à valeurs discrètes. Les histogrammes sont divisés en barres dont la hauteur correspond à la contribution de chaque sous-groupe.

NB : Les options GROUP et SUBGROUP ne sont valables que pour HBAR, VBAR et BLOCK.

4.2.6 proc DOCUMENT : manipulation des ODS

L'ODS DOCUMENT, nouveauté *SAS9*, permet de sauvegarder la structure de vos sorties ODS. La procédure DOCUMENT donne ensuite la possibilité d'organiser cette structure et de recréer des rapports dans différents formats (html, pdf, rtf, etc...) sans avoir à exécuter à nouveau vos programmes *SAS* (cf. exemple 18).

Syntaxe :

```
1ère étape : création des éléments pouvant constituer le rapport.
        ODS DOCUMENT NAME=nombib.nomdoc(write);
            /* Programme SAS */
        ODS DOCUMENT CLOSE;

2ème étape : création du document final.
        PROC DOCUMENT NAME=nombib.nomdoc;
            /* Génération du rapport*/
        RUN;
        ODS format FILE="nomfichier";
            REPLAY;
        RUN;
        ODS format CLOSE;
        QUIT;
```

où *format* est le format désiré : html, ps, rtf, etc...

Exemple 18

```
LIBNAME ex "N:\ProjetStat";
ODS DOCUMENT NAME=ex.essai(write);
    /*création du document essai dans la bibliothèque ex*/
    /*le fichier essai.sas7bitm apparaît sur "N:\ProjetStat"*/

PROC GLM DATA=sashelp.class;
    CLASS sex age;
    MODEL weight=sex age;
RUN;
PROC GCHART DATA=sashelp.class;
    VBAR3D sex / SUBGROUP=age;
RUN;
QUIT;
ODS DOCUMENT CLOSE;

PROC DOCUMENT NAME=ex.essai ;
    DIR glm\data#1;
    HIDE classlevels#1, nobs#1;
        /*on masque les éléments classlevels et nobs*/
ODS rtf FILE="affichage.rtf"  PATH="Q:\math_enseignement\PolysMath\SAS";
        /*le document affichage.rtf est généré à l'adresse du path*/
    REPLAY \glm, \gchart / DEST=rtf;
        /*les résultats des proc glm et gchart sont mis dans le rapport*/
RUN;
ODS rtf CLOSE;
QUIT;
```

On obtient alors le rapport illustré par la figure 5.

The GLM Procedure

Dependent Variable: Weight

Source	DF	Somme des carrés	Carré moyen	Valeur F	Pr > F
Model	6	7206.751155	1201.125193	6.77	0.0026
Error	12	2128.985687	177.415474		
Corrected Total	18	9335.736842			

R-carré	Coeff Var	Racine MSE	Weight Moyenne
0.771953	13.31624	13.31974	100.0263

Source	DF	Type I SS	Carré moyen	Valeur F	Pr > F
Sex	1	1681.122953	1681.122953	9.48	0.0096
Age	5	5525.628202	1105.125640	6.23	0.0045

Source	DF	Type III SS	Carré moyen	Valeur F	Pr > F
Sex	1	862.880980	862.880980	4.86	0.0477
Age	5	5525.628202	1105.125640	6.23	0.0045

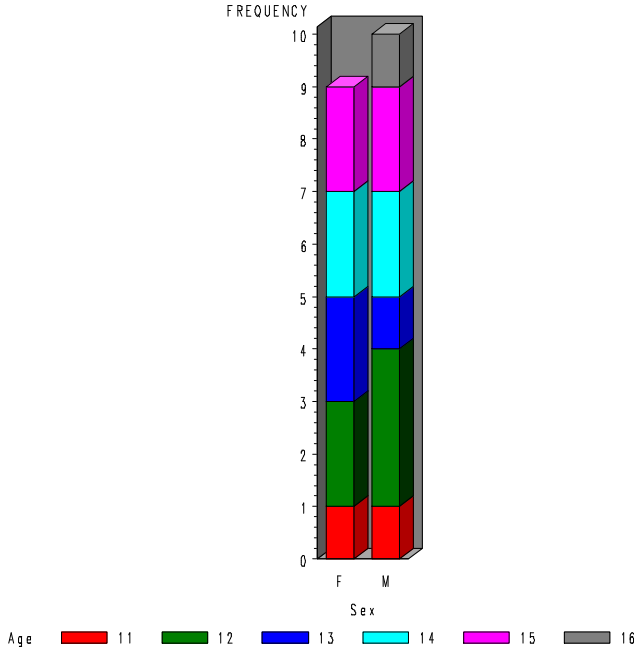

FIGURE 5 – Rapport obtenu par la procédure DOCUMENT

4.3 Les procédures de statistiques descriptives

4.3.1 proc CORR : calcul des corrélations

Cette procédure calcule les coefficients de corrélation (par défaut : coefficient de corrélation de Pearson) entre toutes les variables numériques de la table et édite des statistiques simples.

Syntaxe :

```
PROC CORR options;
    BY variables;
    FREQ variable;
    PARTIAL variables;
    VAR variables;
    WEIGHT variable;
    WITH variables;
RUN;
```

Les instructions :

- PARTIAL donne la liste des variables de contrôle permettant de calculer les coefficients de corrélation partielle de Pearson, Spearman ou Kendall,
- WITH est la liste des variables avec lesquelles sont calculées les corrélations deux à deux des variables VAR.

Quelques options :

- SPEARMAN calcule le coefficient de corrélation des rangs de Spearman,
- KENDALL calcule le coefficient τ de Kendall,
- NOSIMPLE supprime l'impression des statistiques simples (moyenne, somme, min et max, etc...),
- COV édite la matrice des covariances (incompatible avec les options SPEARMAN ou KENDALL),
- OUTP=*nomtab* crée une table *SAS nomtab* contenant les coefficients de corrélation de Pearson.

4.3.2 proc FREQ : tableau croisé, fréquence

Cette procédure permet de construire des tables représentant des distributions statistiques à une variables ou à plusieurs variables qualitatives (tableaux de contingence).

Syntaxe :

```
PROC FREQ DATA=nomtab;
    BY variables;
    TABLES nomtab / options;
    WEIGHT variable;
    OUTPUT OUT=nomtab;
RUN;
```

Les instructions :

- TABLES permet d'obtenir (cf. exemple 19) :
  - une table à une entrée donnant la distribution d'une variable,
  - une table croisant deux variables ou plus, les noms de ces variables sont alors séparés par des astérisques (*),
- OUTPUT OUT=*nomtab*, crée une table *SAS* contenant toutes les statistiques calculées par la procédure FREQ.

Quelques options :

- LIST édite sous forme de listes au lieu de tables (incompatible quand des tests statistiques ou des mesures de liaison sont demandés),
- CHISQ effectue un test du χ^2,
- EXPECTED calcule les fréquences attendues sous l'hypothèse d'indépendance,
- OUT=*nomtab*, crée une table contenant les valeurs des variables et leurs fréquences, pour chaque procédure TABLES (contrairement à l'instruction OUTPUT OUT=*nomtab*, qui ne travaille qu'avec la dernière instruction TABLES).

Exemple 19

```
PROC FREQ;
    TABLES A B C;       /* édition de 3 tables à une entrée         */
RUN;

PROC FREQ;
    TABLES A*B A*C;     /* édition de 2 tables croisées A*B et A*C  */
RUN;

PROC FREQ;
    TABLES A*B*C;       /* pour chaque modalité de A, édition d'une */
RUN;                    /* table croisée B*C                        */
```

4.3.3 proc MEANS : moyennes

Cette procédure calcule des statistiques descriptives univariées (cf. exemple 20).

Syntaxe :

```
PROC MEANS options mots-clés1;
    VAR variables;
    CLASS variables;
    FREQ variable;
    WEIGHT variable;
    ID variables;
    BY variables;
    OUTPUT OUT=nomtab mots-clés2;
RUN;
```

Les instructions :

- CLASS spécifie les variables utilisées pour définir les sous-groupes. Les variables peuvent être numériques ou caractères, mais possèdent un petit nombre de modalités distinctes.
- ID permet d'ajouter la valeur maximale de la variable ID dans la table *nomtab* créée en sortie.
- OUTPUT OUT=*nomtab* définit le nom de la table créée par la procédure.

Quelques options :

- DATA=*nomtab* indique la table *SAS* en entrée,
- NOPRINT supprime toute édition,
- VARDEF=DF, N, WDF ou WGT, indique le diviseur utilisé pour le calcul des variances et covariances :
  - DF : degré de liberté,
  - N : nombre d'observation,
  - WDF : somme des poids - 1,
  - WGT : somme des poids.

Mots-clés de l'instruction Proc Means. Pour obtenir des statistiques supplémentaires :

- N, nombre d'observations n'ayant pas de valeur manquante,
- NMISS, nombre d'observations ayant une valeur manquante,
- MIN, valeur minimale,
- MAX, valeur maximale,
- RANGE, étendue,
- SUM, somme,
- SUMWGT, somme des valeurs de la variable WEIGHT,
- MEAN, moyenne,
- CSS, somme des carrés des écarts à la moyenne,
- USS, somme des carrés,
- VAR, variance,
- STD, écart-type,
- STDERR, erreur type de la moyenne,
- CV, coefficient de variation,
- SKEWNESS, coefficient d'asymétrie,
- KURTOSIS, coefficient d'aplatissement,
- T pour un test t sur la moyenne.

Par défaut, *SAS* édite le nom de la variable, N, MEAN, STD, MIN et MAX.

Mots-clés de l'instruction Output. Les mots-clés correspondent aux statistiques que l'on veut stocker dans la table de sortie. Ils sont à choisir parmi les mots-clés cités ci-dessus.

Exemple 20

```
PROC MEANS DATA=eleves;
    CLASS sexe;
    VAR dictee calcul;
    OUTPUT OUT=resultat MEAN=moy_dictee moy_calcul
                        STD=ect_dictee ect_calcul;
RUN;
```

Remarques :

- *sexe définit le sous-groupe,*
- *les stats sont calculées pour les variables dictee et calcul,*
- *resultat est la table de sortie,*
- *la table resultat contient 4 variables :*
  moy_dictee : moyenne de la variable dictée,
  moy_calcul : moyenne de la variable calcul,
  ect_dictee : écart-type de dictée,
  ect_calcul : écart-type de calcul.

4.3.4 proc UNIVARIATE : fractiles

La procédure UNIVARIATE produit des statistiques univariées classiques pour des variables numériques (moyenne, somme, variance), mais aussi :

- des détails sur les valeurs extrêmes,
- des quantiles,
- des tableaux d'effectifs,
- des graphiques,
- des tests de nullité de la moyenne,
- des tests de normalité de la distribution (coefficient d'asymétrie et d'aplatissement).

Syntaxe :

```
PROC UNIVARIATE options;
    VAR variables;
    BY variables;
    FREQ variable;
    HISTOGRAM variables / options_histo;
    QQPLOT variables / options_qqplot;
    WEIGHT variable;
    ID variables;
    OUTPUT OUT=nomtab mots-clés;
RUN;
```

Les instructions :

- HISTOGRAM crée des histogrammes des variables listées (cf. exemple 21),
- QQPLOT crée des graphes qqplot (Quantile x Quantile Plot) des variables listées,
- ID : la première variable de la liste sert à identifier les observations dans la table des valeurs extrêmes (les 5 plus grandes et les 5 plus petites valeurs),
- OUTPUT OUT=*nomtab* définit le nom de la table créée par la procédure.

Quelques options de l'instruction UNIVARIATE :

- DATA=*nomtab* indique la table *SAS* en entrée,
- FREQ produit une table de fréquence et de pourcentage,
- PLOT permet l'impression de graphiques tige-feuille (steam-and-leaf), de box-plot, et de droites de Henry (normal probability plot).
- NORMAL effectue un test de normalité,
- NOPRINT supprime toutes les éditions.

Quelques options de l'instruction HISTOGRAM et QQPLOT : NORMAL ajoute une courbe selon une distribution normale, CTEXT spécifie la couleur du texte, etc...(voir aide *SAS*)

Mots-clés de l'instruction Output. On retrouve entre-autre, les mêmes mots-clés que ceux de la procédure Means (cf. chapitre 4.3.3).

Exemple 21

```
PROC UNIVARIATE DATA=Feuille;
    HISTOGRAM longueur /    normal
                            ctext     = blue
                            midpoints = 5.6 5.8 6.0 6.2 6.4 ;
                            /*les points moyens des barres de l'histogramme*/
RUN;
```

4.4 La régression linéaire : proc REG

La procédure REG (cf. exemples 22 et 23) est la procédure générale de régression linéaire pour des variables continues. Il en existe d'autres pour des applications plus spécifiques : CATMOD, GENMOD, GLM, LOGISTIC, MIXED, NLIN, PROBIT, etc...

Syntaxe :

```
PROC REG DATA=nomtab;
    MODEL variables_expliquées = variables_explicatives /options;
    VAR variables;
    BY variables;
    FREQ variable;
    WEIGHT variable;
    OUTPUT OUT = nomtab_sortie P=predite R=residu;
    PLOT yvariable*xvariable = symbol
     ...yvariable*xvariable = symbol /options;
RUN;
```

Les instructions :

- MODEL spécifie les variables expliquées et les variables explicatives dans le modèle de regression. On peut définir plusieurs modèles. Cette instruction est obligatoire et possède différentes options.
- OUTPUT OUT=*nomtab* définit le nom de la table créée par la procédure.

- PLOT génère des graphiques traditionnels qui sont indépendants de ceux générés par l'ODS Graphics, disponible depuis la version 9 (cf. chapitre 2.12 et exemple 10).

Quelques options de l'instruction MODEL :

- SELECTION=*méthode* spécifie la méthode de sélection de modèle pour la régression multiple. Par défaut, la méthode de sélection de variables est le modèle complet. Méthode peut prendre différentes valeurs :
  - BACKWARD est une régression pas à pas descendante,
  - FORWARD est une méthode pas à pas ascendante,
  - STEPWISE est une méthode pas à pas ascendante mais avec remise en cause des variables déjà introduites.
- R permet d'obtenir l'analyse des résidus,
- SLENTRY et SLSTAY sont les seuils de signification des tests F d'admission et d'élimination des variables,
- DW calcul la statistique de Durbin-Watson qui permet de tester l'autocorrélation du premier ordre des résidus. Ce test n'a de sens que si les observations sont ordonnées selon une variable.
- COVB estime la matrice de covariance des paramètres estimés,
- CORRB calcule la matrice de corrélation des paramètres estimés.

Quelques options de l'instruction PLOT : On retrouve les mêmes options que celles de la procédure PLOT ou GPPLOT (cf. chapitre 4.2.4) : overlay, haxis, vaxis, href, vref, etc...Mais également :

- CONF trace l'intervalle de confiance à $100(1-\alpha)\%$
- PRED trace l'intervalle de prédiction à $100(1-\alpha)\%$

Il est également intéressant de visualiser la structure des résidus et de tracer le graphe :

```
PLOT residual. * predicted.;
```

Exemple 22

```
PROC REG DATA=sashelp.class;
    MODEL weight=height;
    PLOT r.*p. /cframe=ligr;    /*le fond du graphe est gris clair*/
RUN;
QUIT;
```

```
                                        15:18 Monday, June 12, 2006   1

                          The REG Procedure
                            Model: MODEL1
                       Dependent Variable: Weight

                  Number of Observations Read          19
                  Number of Observations Used          19
```

Analyse de variance

Source	DF	Somme des carrés	Carré moyen	Valeur F	Pr > F
Model	1	7193.24912	7193.24912	57.08	<.0001
Error	17	2142.48772	126.02869		
Corrected Total	18	9335.73684			

Root MSE	11.22625	R-Square	0.7705
Dependent Mean	100.02632	Adj R-Sq	0.7570
Coeff Var	11.22330		

Résultats estimés des paramètres

Variable	DF	Résultat estimé des paramètres	Erreur std	Valeur du test t	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

On obtient alors le graphique 6.

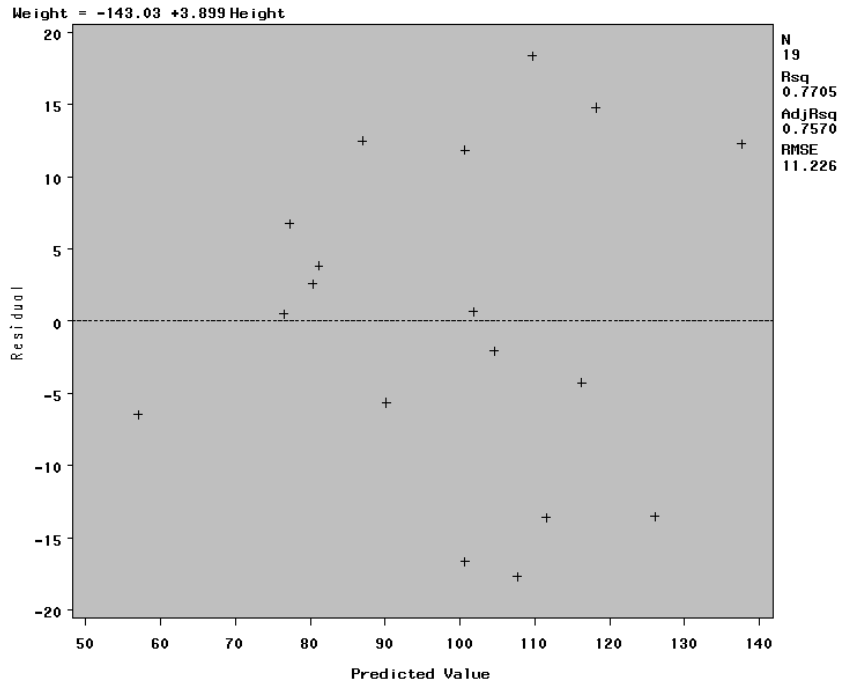

FIGURE 6 – Structure des résidus

Exemple 23

```
DATA rendble;
    INFILE 'ble.don';
    INPUT annee rdt thiver tete pluie;

/* régression avec analyse des résidus et test de Durbin-Watson */
/* création de la table dessin pour 2 graphiques superposés     */
PROC REG DATA=rendble;
    MODEL rdt=thiver tete pluie /R DW;
    OUTPUT OUT=dessin p=rdtp;

PROC GPLOT;
    PLOT rdtp*annee='+' rdt*annee='*' /OVERLAY;

/* régression multiple avec sélection de variables */
PROC REG DATA=renble;
    MODEL rdt=thiver tete pluie /SELECTION=stepwise;
RUN;
QUIT;
```

4.5 Le modèle linéaire général : proc GLM

La procédure GLM (cf. exemple 24) est la procédure générale pour le modèle linéaire. Elle est composée des méthodes statistiques suivantes :

- regression,
- analyse de variance,
- analyse de covariance.

Syntaxe :

```
PROC GLM DATA=nomtab ;
    MODEL variables_dependantes=effets  / options ;
    CLASS variables ;
    BY variables ;
    FREQ variable ;
    WEIGHT variable ;
    LSMEANS effets / options ;
    MEANS effets / options ;
    OUTPUT OUT=nomtab_sortie P=predite R=residu ;
RUN;
```

Les instructions :

- MODEL est l'instruction qui définit le modèle. Les sommes de carrés de type I et III sont calculées par défaut. Cette instruction est obligatoire et possède différentes options :
  - SOLUTION permet d'obtenir en plus de l'analyse de variance, les estimations des paramètres du modèle,
  - P édite les valeurs observées, prédites et les résidus,

- CLASS définit les variables qualitatives,
- LSMEANS calcule la moyenne ajustée de chaque effet listé. On peut ajouter les options :
  - TDIFF et PDIFF pour obtenir les valeurs des statistiques de test d'égalité deux à deux des moyennes ajustées et les probabilités critiques associées,
  - COV pour obtenir les variances et covariances des moyennes ajustées dans la table de sortie,
- MEANS permet de faire des comparaisons de moyennes où *effets* est la liste des effets (ne contenant que des variables déclarées dans CLASS). Les options de MEANS sont entre-autre :
  - BON pour la méthode de Bonferroni,
  - DUNCHAN pour la méthode de Duncan,
  - SCHEFFE pour la méthode de Scheffé,
- OUTPUT OUT=*nomtab_sortie* définit le nom de la table créée par la procédure. P=*predite* est la liste des noms des variables qui contiendront les valeurs prédites et R=*residu* celle des résidus.

Exemple 24

```
DATA test_medic;
    INPUT medicament $ PreTraitement PostTraitement @@;
    CARDS;
A 11  6   A  8  0   A  5  2   A 14  8   A 19 11
A  6  4   A 10 13   A  6  1   A 11  8   A  3  0
D  6  0   D  6  2   D  7  3   D  8  1   D 18 18
D  8  4   D 19 14   D  8  9   D  5  1   D 15  9
F 16 13   F 13 10   F 11 18   F  9  5   F 21 23
F 16 12   F 12  5   F 12 16   F  7  1   F 12 20
;

PROC GLM;
    CLASS medicament;
    MODEL PostTraitement = medicament PreTraitement / solution;
    LSMEANS medicament / stderr pdiff cov out=MoyenneAjustee;
RUN;
QUIT;
```

```
                                        15:18 Monday, June 12, 2006   1

                        The GLM Procedure

                       Informations sur le
                        niveau de classe

          Classe          Niveaux    Valeurs
          medicament            3    A D F

       Number of Observations Read          30
       Number of Observations Used          30
```

Dependent Variable: PostTraitement

Source	DF	Somme des carrés	Carré moyen	Valeur F	Pr > F
Model	3	871.497403	290.499134	18.10	<.0001
Error	26	417.202597	16.046254		
Corrected Total	29	1288.700000			

R-carré	Coeff Var	Racine MSE	PostTraitement Moyenne
0.676261	50.70604	4.005778	7.900000

Source	DF	Type I SS	Carré moyen	Valeur F	Pr > F
medicament	2	293.6000000	146.8000000	9.15	0.0010
PreTraitement	1	577.8974030	577.8974030	36.01	<.0001

Source	DF	Type III SS	Carré moyen	Valeur F	Pr > F
medicament	2	68.5537106	34.2768553	2.14	0.1384
PreTraitement	1	577.8974030	577.8974030	36.01	<.0001

Paramètre		Estimation	Erreur standard	Valeur du test t	Pr > |t|
Intercept		-0.434671164 B	2.47135356	-0.18	0.8617
medicament	A	-3.446138280 B	1.88678065	-1.83	0.0793
medicament	D	-3.337166948 B	1.85386642	-1.80	0.0835
medicament	F	0.000000000 B	.	.	.
PreTraitement		0.987183811	0.16449757	6.00	<.0001

Least Squares Means

medicament	Post Traitement LSMEAN	Erreur standard	Pr > |t|	Nombre LSMEAN
A	6.7149635	1.2884943	<.0001	1
D	6.8239348	1.2724690	<.0001	2
F	10.1611017	1.3159234	<.0001	3

Least Squares Means for effect medicament
Pr > |t| for H0: LSMean(i)=LSMean(j)

Dependent Variable: PostTraitement

i/j	1	2	3
1		0.9521	0.0793
2	0.9521		0.0835
3	0.0793	0.0835	

Les différents modèles : Soit A et B, des variables qualitatives (donc déclarées derrière CLASS), X une variable quantitative et Y une variable quantitative dépendante, alors on peut utiliser les modèles suivants :

```
Y=A          ANOVA à 1 facteur,
Y=A B A*B    ANOVA à 2 facteurs avec interaction,
Y=X          régression simple,
Y=A X        analyse de la covariance,
Y=A X A*X    analyse de la covariance avec interaction.
```

La procédure ANOVA. Cette procédure a la même syntaxe d'instruction que la proc GLM. Elle est plus rapide, mais n'effectue que des analyses de variance pour des plans équilibrés, des anova à un facteur, certains plans en blocs incomplets partiellement équilibrés, des dispositifs en carré latin, et des plans hiérarchiques.

4.6 L'analyse en composantes principales : proc PRINCOMP

La procédure PRINCOMP permet de réaliser une Analyse en Composantes Principales (ACP) et d'obtenir des valeurs propres, des vecteurs propres et les coordonnées des individus dans les composantes principales.

Syntaxe :

```
PROC PRINCOMP DATA=nomtab options;
    BY variables;
    FREQ variable;
    VAR variables;
    WEIGHT variable;
RUN;
```

Quelques options :

- COV permet de réaliser une analyse non normée (par défaut, elle l'est).
- N=n spécifie le nombre n de composantes principales à calculer (égal au plus au nombre de variables analysées). Par défaut, il y aura autant de composantes que de variables.
- PREFIX=*préfixe* spécifie le préfixe utilisé pour générer le nom des n composantes principales. Par défaut, elles s'appelleront PRIN1, PRIN2....PRINn.
- OUT=*nomtab_sortie1* crée une table *nomtab_sortie1* contenant les données de la table analysée ainsi que les coordonnées des individus dans les composantes principales. Cette table est à créer pour :
  - éditer les coordonnées des individus dans les composantes principales,
  - réaliser ensuite des graphiques.
- OUTSTAT=*nomtab_sortie2* crée une table *nomtab_sortie2* contenant les moyennes, les écarts-types, le nombre d'observations, les corrélations ou les covariances, les valeurs propres et les vecteurs propres.

Le tracé des graphiques : La procédure PRINCOMP n'édite aucun graphique par défaut. Un programme commenté *ExACP.sas* associé à une macro *macro-acp.sas* fonctionne sur les données du fichier *Etat.don* . Cet exemple porte sur le budget de l'état français entre 1872 et 1971. Il est disponible sur le volume TD du serveur étudiant Portia :

portia
td
math
Macros_sas
Il permet, en plus des résultats de l'analyse, d'obtenir deux types de graphiques pour lesquels la démarche suivie est rapidement décrite. Il pourra (après l'avoir copié dans un répertoire personnel) être modifié pour s'adapter aux données à traiter.

On peut également réaliser des programmes simplifiés sans macro qui produisent la projection des individus sur les premiers axes principaux (cf exemple 25).

Exemple 25

```
DATA budget;
    INFILE 'T:\math\Macros_Sas\ACP\Etat.don';
    INPUT An$ Pvp Agr Cmi Tra Log Edu Acs Aco Def Det Div;
PROC PRINCOMP OUT=sortie;
TITLE "Budget de l'état entre 1872 et 1971";
PROC PRINT;
RUN;
ODS RTF body='N:\ProjetStat\SortieAcp.rtf';
DATA an;
    SET sortie;
    x=prin1;            /*prin1 et prin2 sont les variables  */
    y=prin2;            /*affectées à x et y                 */
    XSYS='2';           /*Paramètre de mise en forme         */
    YSYS='2';           /*Paramètre de mise en forme         */
    FUNCTION='label';   /*Pour mettre du texte dans le graphe*/
    TEXT=An;            /*Les points sont identifiés par "an"*/
    KEEP x y xsys ysys function text;
    SYMBOL I=none;
    AXIS1 LABEL=('axe1') LENGTH=15cm;
    AXIS2 LABEL=('axe2') LENGTH=10cm;
    TITLE 'budget axe 1 et 2';
PROC GPLOT ANNOTATE=an;
    PLOT y*x /VREF=0 HREF=0 HAXIS=axis1 VAXIS=axis2;
RUN;
QUIT;
ODS RTF CLOSE;
```

Remarque :

- *ANNOTATE=nomtab spécifie la table et toutes ses caractéristiques qui seront utilisées par la procédure GPLOT.*
- *L'ODS (depuis la version8) permet de rediriger les sorties (ici le graphe de la procédure GPLOT) vers un fichier externe.*

On obtient alors le graphique 7.

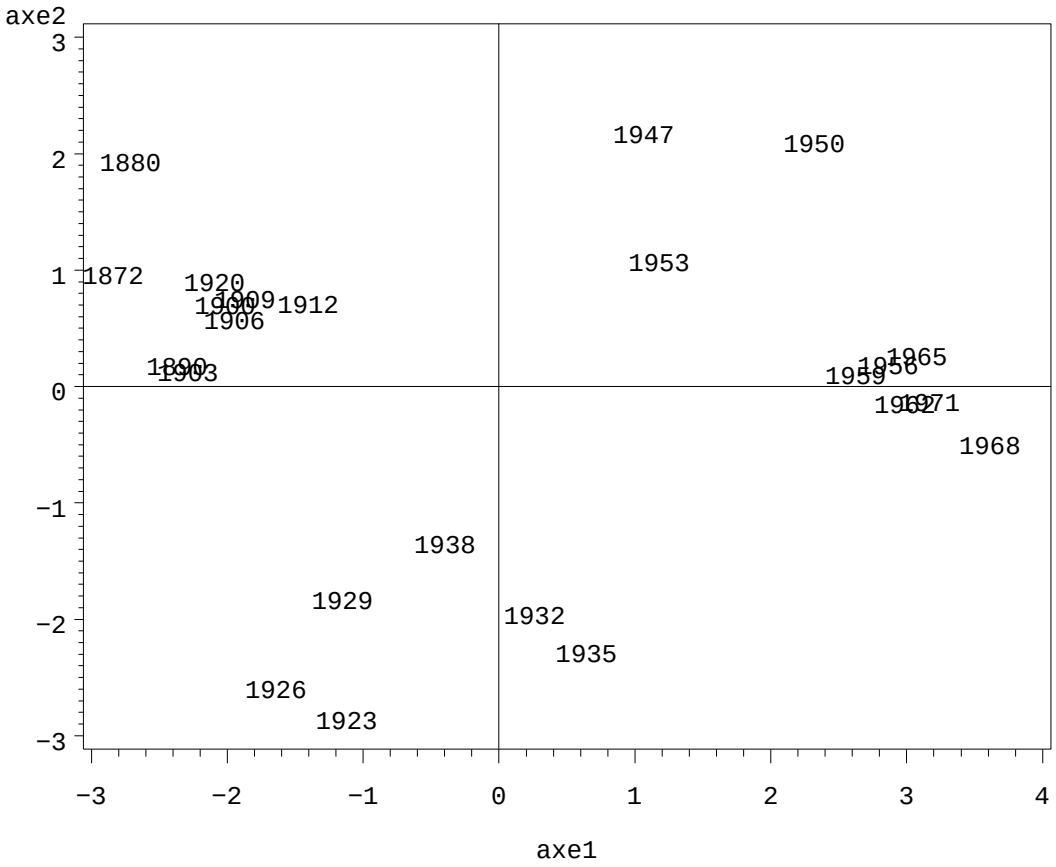

FIGURE 7 – Projection des individus sur les axes 1 et 2

Utilisation de l'ODS. Depuis la version 9, on peut utiliser l'ODS Graphics dans la procédure PRINCOMP, mais c'est expérimental ! (cf. exemple 26)

Exemple 26

```
ODS RTF body='N:\ProjetStat\SortieAcp.rtf';
ODS graphics on;

DATA budget;
    INFILE 'T:\math\Macros_Sas\ACP\Etat.don';
    INPUT An$ Pvp Agr Cmi Tra Log Edu Acs Aco Def Det Div;
PROC PRINCOMP DATA=Budget n=3;
RUN;

ODS GRAPHICS OFF;
ODS RTF CLOSE;
```

On obtient alors le graphique 8.

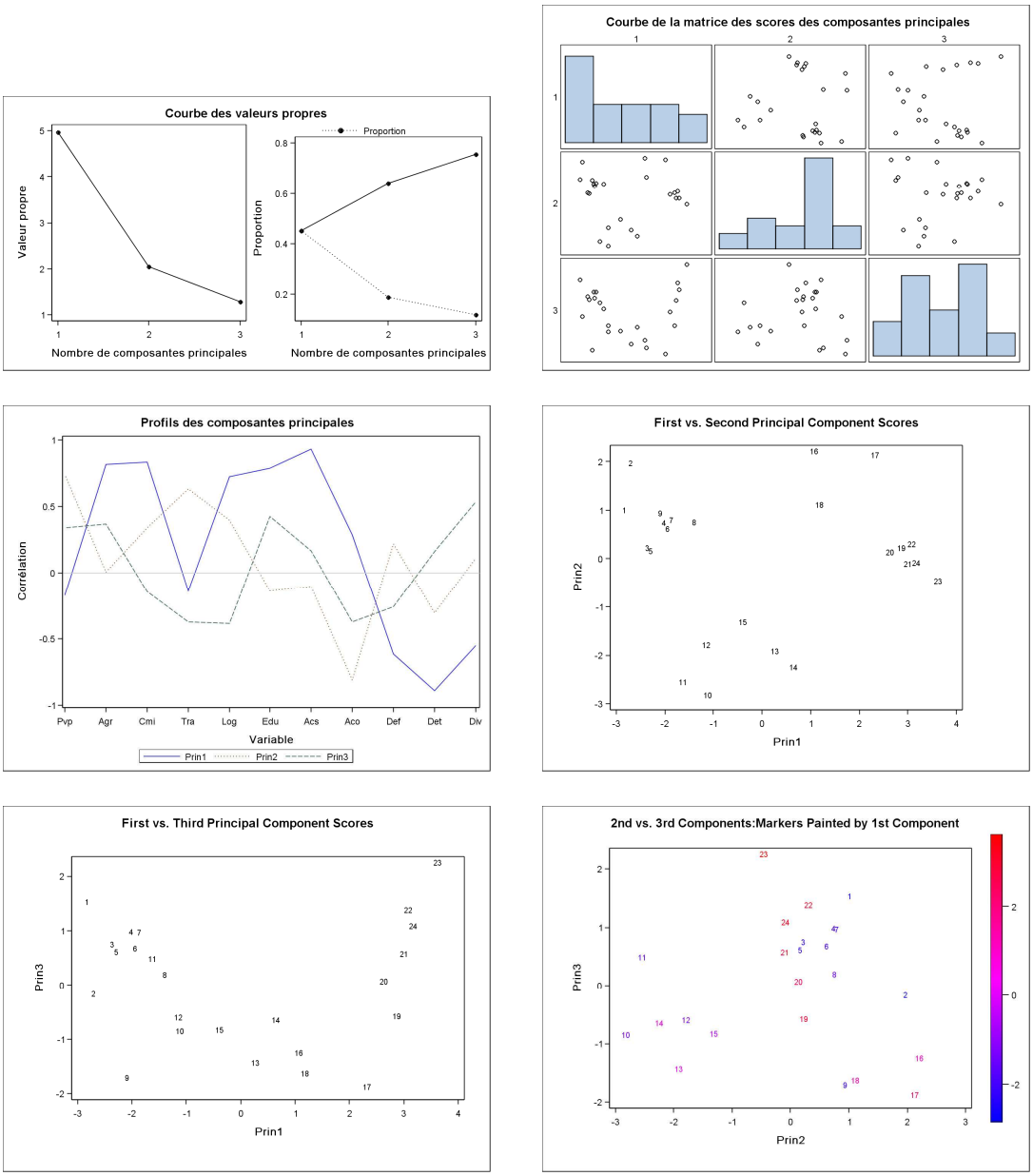

FIGURE 8 – Graphes de la proc PRINCOMP avec l'ODS Graphics

Références

- Introduction au logiciel SAS de base, Support de cours SAS Institute.
- Initiation au logiciel statistique SAS sous UNIX, Catherine Dervin, version septembre 2001.
- SAS sous UNIX Logiciel Hermétique pour Système Ouvert, J.M. AZAÏS, PHILIPPE BESSE, HERVE CARDOT, VINCENT COUALLIER, ALAIN CROQUETTE, version septembre 2001, mises à jour : www.lsp.ups-tlse.fr/Besse