# INTRODUCTION AU LOGICIEL SAS

Julien JACQUES
http://eric.univ-lyon2.fr/~jjacques/

# Table des matières

# 1 Préliminaires

Remarque : il est possible d' exécuter un programmes SAS sans lancer l'interface graphique SAS. Rendez-vous section 2.5 pour plus d'information.

Cette documentation a pour objectif de vous introduire brièvement le logiciel SAS et son utilisation (sous Linux). Pour aller plus loin avec SAS, une documentation très intéressante est disponible à l'adresse suivante :

# Références

[1] J-M. Azaïs, P. Besse, H. Cardot, V. Couallier, A. Croqette.
SAS sous UNIX, Logiciel Hermétique pour Sytème Ouvert.
Version 2001, disponible sur `http://www.math.univ-toulouse.fr/~besse/enseignement.html`

Téléchargez cette documentation et gardez-la sous la main, cela vous sera utile par la suite !

# 2 Introduction à SAS

Le système SAS est un ensemble de modules logiciels pour la gestion et le traitement statistique des données. Suivant les utilisateurs, deux approches peuvent être envisagées :

- l'écriture de *programmes SAS* qui exécutent (entre autre) la gestion des données (importation, saisie...), des analyses statistiques... Les programmes de base que nous utiliserons dans ces TP consisteront à faire appel à une procédure SAS prédéfinies en lui passant les paramètres d'entrées nécessaires.
- l'utilisation d'un système de menu, ou fenêtrage (*solutions SAS*), qui permet de réaliser les procédures statistiques les plus courante, de façon plus conviviale que par l'écriture d'un programme SAS. Mais toutes les procédures SAS ne peuvent être utilisées via ce fenêtrage.

La seconde approche étant plus facile à manipuler, nous ne la verrons pas dans ce TP. Notre objectif sera d'apprendre à écrire et exécuter des programmes SAS. Une fois que vous maîtriserez la manipulation du logiciel SAS et l'écriture de programmes SAS, vous verrez qu'il vous sera très facile d'utiliser les *solutions SAS* si besoin.

## 2.1 Les différentes fenêtres

Lorsque vous lancez SAS, 5 fenêtres s'ouvrent :

- SAS : Explorer
  affiche l'arborescence des librairies (répertoires) et tables gérées par SAS.
- SAS : Results
  permet de gérer l'ensemble des résultats (textes et graphiques) de façon arborescente.
- SAS : Output
  affiche tous les résultats (texte) produits par l'exécution des différentes procédures. Les graphiques apparaissent dans une fenêtre spécifique.
- SAS : Log
  affiche le compte rendu de la bonne exécution et les messages d'erreur.
- SAS : Programm Editor
  est un éditeur de texte rudimentaire pour entrer et modifier les programmes SAS avant d'en demander l'execution.
  Quelques commandes :
  - pour insérer $n$ lignes, tapez dans la zone des numéros de ligne : `in`
  - pour supprimer une ligne, tapez dans la zone des numéros de ligne : `d`
  - pour supprimer un bloc de ligne, tapez dans la zone des numéros de ligne à la première et dernière ligne : `dd`
  - passer du mode insertion au mode suppression : `Ctrl x`

  *Remarque :* ***il est conseillé d'écrire les programmes SAS dans un éditeur de texte LINUX classique (xemacs ou autre), et de le soumettre à l'éditeur SAS par simple copie de la souris***.

Le menu **Run** de la fenêtre *Programm Editor* permet de lancer l'exécution du programme (*submit*) contenu dans l'éditeur, mais également de rappeler dans l'éditeur le programme qui vient d'être exécuté (*recall last submit*).
Enfin, pour quitter SAS, il est conseillé d'utiliser le menu de cette fenêtre.

## 2.2 Librairies et tables SAS

Après saisie ou importation, les données sont gérées par SAS sous la forme d'une **table SAS**.
Une **librairie** est un catalogue de tables SAS.

Nous allons dans un premier temps créer un répertoire dans lequel nous mettrons toutes nos librairies futures

1. créer dans un shell hors de SAS un répertoire qui sera votre répertoire de travail :
   `./TP_Stat_SAS/`
2. créer également un sous-répertoire `Librairies/`, dans lequel figureront toutes les librairies que l'on utilisera sous SAS,

Pour créer une librairie, il faut :

1. la créer dans un shell hors de SAS en tant que répertoire (appelez-la par exemple `tp1_lib`), dans votre répertoire de librairie :
   `mkdir ./TP_Stat_SAS/Librairies/tp1_lib`
2. la déclarer ensuite dans SAS (dans l'éditeur de programme) en tant que libraire. Pour cela, il faut utiliser la commande `libname` comme suit :
   `LIBNAME tp1_lib './TP_Stat_SAS/Librairies/tp1_lib';`

En suivant cette démarche, les tables créées seront permanentes. Vous pourrez les retrouver dans une utilisation ultérieure de SAS. La première chose à faire sera alors de charger la librairie SAS comme ci-dessus à l'aide de la commande `libname`.
Néanmoins, il n'est pas utile de conserver toutes les tables. Il est donc possible d'utiliser des tables temporaires. Pour cela il suffit de ne préciser que le nom de la table lors de sa création, sans lui affecter une librairie. La librairie par défaut utilisée sera la librairie *WORK*, et son contenu sera vidé lorsque vous quitterez SAS.

## 2.3 Programme SAS

Un programme SAS est un enchaînement d'étapes de gestion de données (DATA) et d'appels de procédures (PROC) décrivant les traitements à réaliser sous le couvert d'*options* prises par défaut ou définies explicitement.
Les différentes étapes ou procédures communiquent entre elles uniquement par l'intermédiaire de tables SAS, permanentes ou temporaires, et avec l'extérieur par des tables SAS ou des fichiers textes usuels.

### 2.3.1 Etape DATA

Dans cette étape nous pouvons saisir ou importer des données.

**Saisie des données.** Voici un programme qui créé une table appelée `table1` (dans la librairie `tp1_lib`), contenant les données de 3 individus décrits sur 4 variables.

```
DATA tp1_lib.table1;
   input nom $ sexe $ taille datenaissance;
   format datenaissance DDMMYY10.;
   informat datenaissance DDMMYY10.;
   cards;
   tutu M 170 11-12-82
   toto M 182 21-12-82
   titi F 157 25-12-83
   run;
```

L'instruction `input` indique à SAS les variables qu'il devra lire.
A la suite du nom de la variable, il faut préciser le format : l'absence de format signifie que la variable est numérique, le format $ signifie que la variable est de type alphanumérique (à utiliser pour les variables qualitatives), le

format DDMMYY10. est un des formats de date. Le format peut également être précisé dans une ligne suivante à l'aide de l'instruction `format`.
Remarque sur les formats de date : en interne, tous les formats de type temporel (date, heure, intervalle de temps) sont codés en valeur numérique, afin de faciliter les calculs notamment. SAS convertit donc automatiquement une date en valeur numérique (nombre de jours écoulés depuis le 01/01/1960). Si vous voulez afficher les dates en format date et non en format temporel, il faut lui préciser le format de lecture à l'aide de la commande `informat`.

Les données devront ensuite être saisies à la suite de l'instruction `cards;`, un individu par ligne, sans ";" à la fin de chaque ligne.

La table ainsi créée sera désormais identifiée par `tp1_lib.table1`, et stockée dans `table1.sas7bdat` dans le répertoire `tp1_lib` relatif à la librairie.

**Importation de données.** Nous considérons un fichier de données `donnees1.dat` contenant les données précédentes (fichier texte).
Ce programme importe les données de `donnees1.dat` dans la table `table2` de la librairie `tp1_lib`.

```
DATA tp1_lib.table2;
   infile '/home/gis3/.../Donnees/donnees1.dat' dlm=' ' firstobs=1;
   input nom $ sexe $ taille datenaissance;
   format datenaissance DDMMYY10.;
   informat datenaissance DDMMYY10.;
   run;
```

Par défaut, le fichier texte doit contenir un individu par ligne. Le **séparateur** de variable est indiqué dans le paramètre `dlm` de l'instruction `infile` (ici c'est un espace). Pour une tabulation, indiquez : `dlm="09"x`.
Lorsque le fichier de données contient une **ligne d'entête**, il faut indiquer à l'option `firstobs` à quelle ligne commencer la lecture des observations (2 si le fichier contient une ligne d'entête).
Lorsque le fichier de données contient plusieurs individus par ligne, il faut indiquer à la fin de l'instruction `input` le double caractère @@ qui a pour effet de maintenir un article dans le buffer de lecture jusqu'à ce qu'il soit lu complètement :

```
input nom$ sexe$ taille datenaissance @@;
```

Enfin, si les données sont collées, il est possible d'indiquer à la suite de chaque nom de variable dans l'instruction `input` les positions (numéros de colonne) concernées :

```
input nom$ 1-4 sexe$ 5 taille 6-8 datenaissance 9-16;
```

Pour toute manipulation de données plus avancée, reportez-vous à [1].

**Transformation de données.** La procédure `DATA` permet également de créer de nouvelles variables. Dans le code suivant, la table `tp1_lib.table_in` contenant la variable `var_in` est chargée, puis une nouvelle variable `var_out` est crée à partir de `var_in`. Toutes les variables (nouvelles et anciennes) sont alors enregistrées dans une nouvelle table `tp1_lib.table_out`.

```
DATA tp1_lib.table_out;
   set tp1_lib.table_in;
   var_out = var_in - 4;
   run;
```

**`PROC SQL`** Cette procédure permet une gestion rapide et efficace des bases de données SAS, souvent plus simple qu'en passant par une étape `DATA`. Pour son utilisation, référez-vous à [1] ou à la documentation SAS en ligne.

### 2.3.2 Etape PROC

SAS contient un grand nombre de procédure de base, dont la syntaxe générale est la suivante :

```
PROC nom_procedure data=donnees;
   instructions eventuelles;
   run;
```

La procédure `print` permet d'imprimer dans la fenêtre *output* de SAS le contenu d'une table :

```
PROC PRINT data=tp1_lib.table2;
   run;
```

L'option `(obs = 40)` mis à la suite du nom de la table permet d'imprimer uniquement les 40 premières lignes de cette table.

Reportez vous à l'aide de SAS pour toute question relative à une procédure. Vous pourrez utiliser soit l'aide locale (Menu Help), soit l'aide en ligne suivante (plus pratique) :

`http://support.sas.com/91doc/docMainpage.jsp`

Outre le site officiel de SAS, vous trouverez également sur internet un grand nombre de sites dédiés à ce logiciel. Ayez le réflexe de faire une recherche internet lorsque vous avez une question.

## 2.4 Quelques éléments de programmation sous SAS

Le langage reconnaît les expressions mathématiques usuelles, les fonctions mathématiques usuelles (`round`, `sin`, `log`, `sqrt`,...), les fonctions de gestion de chaînes de caractères (`length`, `scan`, ...), les fonctions à usage plus statistique (`sum`, `mean`, `min`, `max`, `var`, `std`, ...). Ces dernières s'appliquent à une liste de variables avec la syntaxe suivante : `sum(var1, of var3-var5, var8)`

**If-Then.** La syntaxe est la suivante :

```
IF condition THEN action;
```

ou encore

```
IF condition THEN DO;
   action1;
   action2;
END;
```

Pour écrire un commentaire dans un programme SAS, utilisez les caractères /* et */ en début et fin de commentaire :

/* *ceci est une ligne de commentaires* */

Quelques autres commandes qui peuvent être ajoutées au début d'un programme SAS :

- `Options` : ce sont ici les options générales. Parmi celles-ci, `pagesize` spécifie le nombre de lignes dans une page de sortie, `linesize` spécifie le nombre de caractères par ligne, `nodate` supprime l'impression de la date dans les pages de sorties.
- `Title` : place un titre en haut de chaque page de sortie.
- `Footnote` : place un bas de page sur chaque page de sortie.

## 2.5 Exécution différée d'un programme SAS

Il est possible d'exécuter des programmes SAS sans ouvrir le logiciel SAS. Cela peut être utile notamment lorsque les programmes nécessitent un temps d'exécution long. En pratique, cela diminue aussi les ressources demandées à l'ordinateur pour gérer l'affichage graphique des différentes fenêtres SAS.
Pour cela, il suffit d'enregistrer votre programme sous le nom `mon_prog.sas`, et de lancer son exécution à l'aide de la commande suivante dans un terminal :

```
/usr/local/SAS/SASFoundation/9.2/sas mon_prog.sas -fsdevice x11.motif
```

A noter qu'il est nécessaire de s'être au préalable loguer sur le serveur `weppes`.
Les résultats sont alors regroupés dans un fichier `mon_prog.lst` tandis que le compte-rendu de l'exécution ainsi que les messages d'erreurs se trouvent dans le fichier `mon_prog.log`.

# 3 Statistiques élémentaires

Nous décrivons quelques unes des procédures SAS les plus utiles pour décrire des données et faire les tests statistiques les plus courants :

- `means`, `univariate` : description élémentaire de variable quantitative, tests,
- `freq` : description élémentaire de variable qualitative, tests,
- `plot` : réalise le nuage de points relatif à 2 variables quantitatives,
- `chart` : réalise différents graphiques pour une variable qualitative,
- `sort` : range le fichier selon les valeurs croissantes ou décroissantes d'une variable quantitative spécifiée par la commande `by`,
- `corr` : calcul les matrices de corrélations et de variance-covariance d'un ensemble de variables quantitatives.
- `reg` : permet d'effectuer une régression linéaire.
- `glm` : permet de réaliser une analyse de variance.

Remarque : les procédures chart et plot réalisent des graphiques basses résolutions. Reportez-vous à [1] pour des graphiques plus avancés.
Quelques procédures sont détaillées ci-après, pour celles qui ne le sont pas reportez-vous à l'aide de SAS.

## 3.1 PROC UNIVARIATE

Cette procédure permet de faire des analyses statistiques uni-variées de variables quantitatives : indicateur de tendance centrale et de dispersion, autres caractéristiques de la distribution (quantiles, skewness, kurtosis), graphiques (histogrammes, boîte-à-moustaches, droite de Henry) et des tests : test de nullité de la moyenne de Student, test de normalité, quelques tests non-paramétriques (signe, Wilcoxon...).

### 3.1.1 Syntaxe

```
 PROC UNIVARIATE <options>;
var liste de variables;
by variable;
class variable;
where (variable1=5) or (variable2=5);
weight variable;
output <out=table sas> <liste de statistiques>;
```

### 3.1.2 Options

La liste des options permet de préciser les résultats attendus :
- `data = table sas` : indique le nom de la table SAS (par défaut, c'est la dernière crée),
- `normal` : pour faire des tests de normalité,
- `plot` : pour faire des graphiques,
- `mu0=4` : pour rester l'égalité de la moyenne à 4 (0 par défaut).

### 3.1.3 Commandes

- **`var`** : liste des variables concernées (par défaut, toutes les variables quantitatives),
- **`by`** : suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations correspondant aux modalités de la variable qualitative (la table doit être triée),
- **`class`** : suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations correspondant aux modalités de la variable qualitative,
- **`where`** : ne conserve que les individus de la base de donnée dont la variable1 vaut 5 ou la variable2 vaut 6,
- **`weight`** : nom de la variable contenant les pondérations des observations,
- **`output`** : indique le nom de la table SAS de sortie et la liste des statistiques qui y seront enregistrées.
- **`histogram /kernel`** : trace l'histogramme et un estimation par noyau de la densité.

## 3.2 PROC MEANS et SUMMARY

Elles permettent les mêmes analyses que la procédure `univariate`, graphique et tests non-paramètrique en moins.
La présentation des résultats est différente de celle faite par `univariate` (plus lisible).
`summary` et `means` différent entre elles par les options par défaut.

## 3.3 PROC TTEST

Cette procèdure permet d'effectuer des tests de Student sur la moyenne pour un échantillon, mais également des tests de comparaison de moyenne et de variance. Elle fournit également des statistiques de bases dont des intervalles de confiance sur la moyenne et la variance.

### 3.3.1 Syntaxe

```
 PROC TTEST <options>;
var variable;
by variable;
class variable;
freq variable;
weight variable;
paired variable;
```

### 3.3.2 Options

La liste des options permet de préciser les résultats attendus :
- `data = table sas` : indique le nom de la table SAS (par défaut, c'est la dernière crée),
- `alpha=0.1` : fixe le risque de première espèce à 0.1,
- `H0=m` : pour tester l'égalité à m de la moyenne (m=0 par défaut),
- `sides=U` : pour tester une hypothèse alternative unilatéral supérieure (L pour inférieure).

### 3.3.3 Commandes

- **`var`** : liste des variables concernées (par défaut, toutes les variables quantitatives),
- **`by`** : suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations correspondant aux modalités de la variable qualitative (la table doit être triée),
- **`class`** : suivi du nom d'une variable qualitative identifiant les échantillons à comparer, permet de réaliser un test de comparaison des moyennes des échantillons,
- **`freq`** : fréquence d'occurrences de chaque observation,
- **`weight`** : nom de la variable contenant les pondérations des observations,
- **`paired`** : indique le nom des variables à étudier pour un test de comparaison d'échantillon appariés. Les variables doivent être séparées par une étoile (*). Les commandes **`var`** et **`class`** ne peuvent pas être utilisée avec cette commande.

## 3.4 PROC FREQ

Cette procédure traite les variables qualitatives.
Elle fournit des tris à plat et permet des analyses multi-variées de tables de contingences : profils lignes et colonnes, test d'indépendance du $\chi^2$, comparaisons avec les valeurs déduites du modèle d'indépendance.

### 3.4.1 Syntaxe

```
 PROC FREQ <options>;
by variable;
tables liste des croisements requis </options>;
weight variable;
```

### 3.4.2 Options

- `data = table sas` : indique le nom de la table SAS (par défaut, c'est la dernière crée),
- `order = freq` : édition ordonnées des effectifs par ordre croissant.

### 3.4.3 Commandes

- **by** : suivi du nom d'une variable qualitative indique que les statistiques sont calculées par groupe d'observations correspondant aux modalités de la variable qualitative (la table doit être triée),
- **tables** : liste des variables à étudier ou croisements de variables (de la table de contingences) exprimés sous une des formes : $a$, $a * b$, $a * (bc)$, $(ab) * (cd)$, (a–c)*c. Les options indiquées après un slash précisent les analyses statistiques demandées : la plus utile étant `chisq` qui exécute un test d'indépendance du $\chi^2$,
- **weight** : nom de la variable contenant les pondérations des observations.

## 3.5 PROC PLOT ou PROC GPLOT

Cette procédure trace un nuage de points en deux dimensions.
La PROC GPLOT réalise les mêmes graphiques que la PROC PLOT mais de façon plus élégante.

### 3.5.1 Syntaxe

```
 PROC PLOT <options>;
by variable;
plot liste de graphiques</<options>;
```

### 3.5.2 Options

- data = table sas : indique le nom de la table SAS (par défaut, c'est la dernière crée),
- hpercent : part horizontale utilisée de la page,
- hpercent : part verticale utilisée de la page.

### 3.5.3 Commandes

- **by** : suivi du nom d'une variable qualitative indique que les graphiques seront tracés pour les différents groupes d'observations correspondants aux modalités de la variable qualitative (la table doit être triée),
- **plot** : liste des graphes sous la forme : variable_y*variable_x=caractère, avec la même syntaxe que pour les tables de contingences. Sur l'axe des ordonées figurera la variable_y, sur l'axe des abscisses variable_x, et les points seront représentés par le symbole spécifié dans caractère. Il est également possible de donner un nom de variable à caractère, dans ce cas chaque individus sera représenté sur le graphique par sa valeur pour cette variable.

## 3.6 PROC BOXPLOT

Cette procédure permet de tracer un Boxplot relatif à la distribution d'une variable quantitative (var_quanti) pour chaque modalité d'une variable qualitative (var_quali).

### 3.6.1 Syntaxe

```
 PROC BOXPLOT <options>;
plot var_quanti * var_quali;
```

## 3.7 PROC CHART ou PROC GCHART

Cette procédure trace des diagrammes en barres (hbar), en colonnes (vbar), en camembert (pie) et aléatoires (star). Elle traite des variables qualitatives et quantitatives, ces dernières étant regroupées en classes.
La PROC GCHART réalise les mêmes graphiques que la PROC CHART mais de façon plus élégante.

### 3.7.1 Syntaxe

```
 PROC CHART <options>;
by <descending> variable;
vbar liste de variables</<options standards><options spécifiques»;
hbar liste de variables</<options standards><options spécifiques»;
pie liste de variables</<options standards><options spécifiques»;
star liste de variables</<options standards><options spécifiques»;
```

### 3.7.2 Options

- data = table sas : indique le nom de la table SAS (par défaut, c'est la dernière créée),
- lpi : facteur d'échelle (nb lignes par pouce/nb de colonnes par pouce)*10.

### 3.7.3 Commandes

- **by** : suivi du nom d'une variable qualitative indique que les graphiques seront tracés pour les différents groupes d'observations correspondants aux modalités de la variable qualitative (la table doit être triée),
- **vbar** : permet de réaliser un histogramme avec des barres verticales, - **pie** : permet de réaliser un histogramme en cammenbert, - **star** : permet de réaliser un histogramme en étoile.

### 3.7.4 Options standards

- sumvar : variable quantitative dont le cumul ou la moyenne est représenté,
- freq : variables de pondération des observations,
- midpoints : liste des bornes de classes,
- levels : nombres 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=).

### 3.7.5 Options spécifiques

- group= : représentation de plusieurs graphes côte-à-côte selon les modalités de la variables spécifiées,
- subgroup= : découpage des barres ou colonnes selon les modalités de la variables spécifiées.

De nombreuses autres options sont disponibles, consultez l'aide si besoin.

## 3.8 PROC SORT

Cette procédure permet de trier une table de données.

### 3.8.1 Syntaxe

```
 PROC SORT <options>;
by variable1;
```

### 3.8.2 Options

- `data = table sas` : indique le nom de la table SAS (par défaut, c'est la dernière créée),
- `out = table sortie` : indique le nom de la table SAS de sortie.

### 3.8.3 Commandes

- **by** : suivi du nom d'une variable permet de trier suivant la variable1 (ordre croissant par défaut). En ajoutant `DESCENDING` entre **by** et le nom de la variable, le tri sera fait par ordre décroissant.

## 3.9 PROC CORR

Cette procédure étudie les liaisons entre variables quantitatives : coefficients de corrélation de Pearson et de Spearman, tests associés...

### 3.9.1 Syntaxe

```
 PROC CORR <options>;
by <descending> variable;
var liste de variables;
weight variable;
with liste de variables;
```

### 3.9.2 Options

- `data = table sas` : indique le nom de la table SAS (par défaut, c'est la dernière créée),
- `hoeffding kendall pearson spearman` : sélectionne les types de mesure de corrélation (`pearson` par défaut).

### 3.9.3 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** : liste des variables concernées (par défaut, toutes les variables quantitatives),
- **with** : liste des variables avec lesquelles les corrélations avec les variables de la liste `var` sont à calculer (par défaut, toutes celles de la liste `var`),
- **weight** : nom de la variable contenant les pondérations des observations.

## 3.10 PROC REG

Cette procédure permet d'effectuer une régression linéaire (simple ou multiple). La description donnée ici considère une régression simple.

### 3.10.1 Syntaxe

```
 PROC REG <options1>;
model variable_y=variable_x </<options2>;
plot liste de graphiques;
test equation;
```

### 3.10.2 Options1

- `data = table_sas` : indique le nom de la table SAS (par défaut, c'est la dernière créée).
- `alpha = 0.05` : pour des intervalles de confiances à 95%.

### 3.10.3 Options2

- `clb` : calcule des intervalles de confiance sur les paramètres estimés.
- `corrb` : calcule la matrice de corrélation des estimateurs des paramètres de la régression.
- `covb` : calcule la matrice de covariance des estimateurs des paramètres de la régression.
- `noint` : effectue une régression passant par l'origine.
- `p` : calcule les valeurs prédites.
- `vif` : calcule les VIF (facteur d'inflation de la variance).
- `r` : calcule les résidus (studentisés et non) ainsi que la distance de Cook.

- `selection = bakcward` : effectue la sélection des variables par recherche descendante. Est aussi disponible `forward`, `stepwise`.
- `selection = rsquare best=1` : effectue la sélection des variables par l'algorithme de Furnival et Wilson. SAS retourne le meilleur modèle à 1 variable, 2 variables ... jusqu'au modèle complet. L'ajoût de la commande suivante permet d'afficher les critères de choix de modèles pour chacun des modèles retournés par Furnival et Wilson.
- `cp aic bic rsquare adjrsq` : évalue les critères de choix de modèles $C_p$, AIC et BIC, $R^2$ et $R^2$ ajusté.

### 3.10.4 Commandes

- **`model`** : permet de déclarer la variable à expliquer `variable_y` et la variable explicative `variable_x`.
- **`plot`** : liste des graphes sous la forme : `variable_y*variable_x=caractère`, avec la même syntaxe que pour les tables de contingences. Sur l'axe des ordonées figurera la `variable_y`, sur l'axe des abscisses `variable_x`, et les points seront représentés par le symbole spécifié dans `caractère`.
- **`test`** : permet d'effectuer des tests sur les parametres. Par exemple **`test`** `variable_x=0` teste si le paramètre de la régression relatif à la variable `variable_x` est nul. Mettre `intercept=0` pour tester l'intercept.

### 3.10.5 Astuce

Si on veut tester un modèle $Y = aX + b$ où un des paramètres est connu, par exemple $b = 4$, on crée une nouvelle variable $Y' = Y - 4$ et on estime la régression $Y' = aX$. De même pour $a = 4$, on crée $Y' = Y - 4X$ et on estime la régression $Y' = b$.

## 3.11 PROC GLM

Cette procédure permet entre autre de réaliser des analyses de variances.
On se référera aux résultats de type III. Les résultats de type I peuvent être utile dans des cas de modèle très particuliers que nous n'abordons pas dans ce cours.

### 3.11.1 Syntaxe

```
 PROC GLM;
class facteur1 facteur2;
model variable = facteur1 facteur2 facteur1*facteur2;
means facteur1 / scheffe;
```

### 3.11.2 Commandes

- **class** : définit les (ou la) variables qualitatives qui auront le rôle de facteurs pour l'analyse de variance.
- **model** : permet de déclarer la variable à expliquer `variable` et les facteurs explicatifs.
- **means** : permet de faire des tests de comparaison de moyenne deux à deux selon la procédure de Scheffé.

## 3.12 PROC ANOVA

La `PROC ANOVA` permet de réaliser également une analyse de variance, lorsque les données sont équilibrées (nombre de mesures identique pour chaque modalité).

### 3.12.1 Syntaxe

```
 PROC ANOVA;
class facteur1 facteur2;
model variable = facteur1 facteur2 facteur1*facteur2;
means facteur1 / hovtest=levene scheffe;
```

### 3.12.2 Commandes

- **class** : définit les (ou la) variables qualitatives qui auront le rôle de facteurs pour l'analyse de variance.
- **model** : permet de déclarer la variable à expliquer `variable` et les facteurs explicatifs.
- **means** : permet de faire un test d'homogénéité des variances à l'aide du test de Levene en précisant la commande `hovtest=levene` après un slash, ou des tests de comparaisons des moyennes deux à deux en indiquant `scheffe`.

## 3.13 PROC LOGISTIC

Cette procédure permet entre autre de réaliser des régressions logistiques.

### 3.13.1 Syntaxe

```
 PROC LOGISTIC <options1>;
class variable_x;
model variable_y=variable_x;
```

### 3.13.2 Options1

- `data = table_sas` : indique le nom de la table SAS (par défaut, c'est la dernière créée).

### 3.13.3 Commandes

- **`class`** : à n'utiliser que pour la régression logistique ordinale. Il faut pour ce faire indiquer dans cette commande quelles sont les variables explicatives `variable_x`.
- **`model`** : permet de déclarer la variable à expliquer `variable_y` et la variable explicative `variable_x`.

La `PROC GENMOD` permet de réaliser également une régression logistique.

## 3.14 PROC NPAR1WAY

Cette procédure permet entre autre de réaliser des tests non paramétriques de comparaison d'échantillon (Wilcoxon, Kruskall-Wallis).

### 3.14.1 Syntaxe

```
 PROC NPAR1WAY <options1>;
class facteur;
var variable;
```

### 3.14.2 Options1

- `data = table_sas` : indique le nom de la table SAS (par défaut, c'est la dernière créée).

### 3.14.3 Commandes

- **`class`** : variable séparant les échantillons à comparer.
- **`var`** : variable sur la base de laquelle les échantillons sont comparés.