TISD – FICHE 3

Une prise en main du logiciel SAS

Adrien Hardy, adrien.hardy@math.univ-lille1.fr

Le système SAS est un ensemble de modules logiciels pour la gestion et le traitement statistique des données. Ils permettent d'exécuter des saisies, importations, et transformations de données ; des analyses statistiques ; de générer des rapports, numériques et graphiques. Un programme SAS est typiquement une suite d'étapes `data` (données) et `proc` (procédures).

L'interface de base se compose des fenêtres suivantes :

- *Explorateur* : affiche l'arborescence des librairies (répertoires) et tables gérées par SAS. NB : on remonte dans l'arbre en appuyant sur la touche retour arrière.
- *Résultats* : affiche l'arborescence des résultats (sorties de procédures, graphiques)
- *Editeur* : éditeur de texte où l'on écrit les programmes SAS avant de les exécuter en appuyant sur "run" ou F3. NB : Contrairement à R, il faut sélectionner une zone pour l'exécuter seule, sinon SAS exécute la totalité du script de l'éditeur
- *Journal* : affiche le compte rendu de l'exécution de chaque programme et les messages d'erreurs éventuels. À surveiller de près !
- *Sortie* : affiche les résultats des procédures exécutés (sauf les graphiques qui s'ouvrent dans une fenêtre spécifique).

Commentaires : En utilisant les balises `/*` et `*/` (ou `*` et `;`) on empêche l'exécution de la zone délimitée.

1 Acquisition de données & librairies

1.1 Saisie de données

1. On commence par créer une *table SAS* (SAS data set) à la main qu'on appelle Table. Pour ce faire, entrer dans l'éditeur le programme `data` suivant et l'exécuter :

```
data Table;
input numero taille poids sexe $ sexecode;
cards;
1 174 65 m 1
2 169 56 f 2
3 166 48 f 2
4 181 80 m 1
```

```
5 168 53 f 2
6 176 76 m 1
7 190 77 m 1
8 159 70 f 2
9 162 60 f 2
10 164 51 f 2
11 160 73 f 2
run;
```

NB : le dollar indique que la variable le précédant est qualitative (caractères alphanumériques).

2. Pour afficher les données, on utilise la procédure `proc print` :

```
proc print data=Table;
run;
```

Pour n'afficher que les trois premières lignes, on écrit à la place `proc print data=Table(obs=3);`

3. Sans précision, une table SAS est toujours créée dans la librairie `work`. Rechercher-la avec l'explorateur et ouvrir la table. On souhaite maintenant placer notre table dans un autre répertoire de chemin donné (par exemple `C:/Docs/TP`). Assigner une nouvelle librairie nommée Malib à ce répertoire en exécutant :

```
libname Malib 'chemin';
run;
```

Reprendre la question **1** en remplaçant `Table` par `Malib.Table` et comprendre ce qu'il se passe.

4. On peut rajouter des options aux procédures. Par exemple, on utilisera l'option `label` de `print` pour rajouter des labels à chaque variables (autrement, les noms de variables ne peuvent pas contenir d'espace et ont 8 caractères au maximum)

```
proc print data=Malib.Table label;
label taille="taille de l'élève" poids="poids de l'élève";
run;
```

Le `label` de la première ligne indique que l'on utilise l'option label ; le second `label` détaille l'utilisation de l'option, ici les labels à utiliser.

1.2 Importation de données

1. Télécharger depuis Moodle le fichier `ozone.xls`, puis l'importer sur SAS dans Malib en utilisant `Fichier>Importer données` (avec pour "Member" : ozone).

2. Entrer

```
proc contents data=Malib.ozone;
run;
```

puis afficher la moitié du jeu de données.

2 Manipulation des données

1. Opérations élémentaires sur les variables numériques :

```
data malib.nombres;
input x y;
cards;
5 5
2 -3
4.5 10
3.2 1
2 0
run;

data malib.calcul;
set malib.nombres;
a=x+y;    b=x-y;    c=x*y;   d=x**y;   e=min(x,y);   f=max(x,y);
g=x/y;    h=abs(y);   i=exp(x);     j=int(x);    k=log(y);
l=log10(x);   m=sign(y);   n=sqrt(x);
run;
```

L'entrée `set` annonce l'on va utiliser des variables dans la table SAS appelée.

2. On peut faire des boucles et simuler des variables aléatoires :

```
data malib.compt;
do i=1 to 100 by 1;
x=rand('binomial', 0.4, 20);
y=1+x;
z=x;
x=x-1;
output malib.compt;
end;
run;
```

Ouvrir la table, remarquer qu'il y a un ordre dans les opérations. Remplacer la loi binomiale par d'autres lois ; voir l'aide sur `rand` pour voir les lois disponibles (cf. Aide/Index).

4. On reprend la table `malib.Table` créée à la Section 1.1. Pour trier cette table suivant, par exemple, la taille des individus, on utilise la procédure `sort` :

```
proc sort data=malib.Table;
by taille;
run;
```

5. Trier la table `malib.Table` par sexe et utiliser la `proc print` avec la commande `by sexe` pour avoir la liste des garçons et la liste des filles.

6. Pour garder des variables d'une table, on utilise `keep` :

```
data malib.Table2;
set malib.Table;
```

```
keep numero taille poids sexecode;
run;
```

7. Pour supprimer des variables, on utilise `drop` :

```
data malib.Table3;
set malib.Table;
drop taille poids;
run;
```

8. Pour reconstituer la table `Table` à partir de `Table2` et `Table3` on utilise `merge` :

```
data malib.Table4;
merge malib.Table2 malib.Table3;
by numero;
run;
```

NB : La ligne `by numero;` assure que les tableaux sont bien regroupés en joignant les lignes associées à une même valeur de la variable `numero` qui nous sert ici d'identifiant. Il est nécessaire que les deux tableaux soient triés préalablement suivant `numero`.

9. On veut ajouter les trois informations suivantes (incomplètes) à la table `malib.Table` :

```
data malib.tab1;
input numero sexecode;
cards;
12 1
13 1
14 2
run;
```

Pour cela, faire :

```
data malib.Table5;
set malib.Table malib.tab1;
run;
```

10. On peut extraire des variables de la façon suivante :

```
data malib.Table6;
set malib.Table;
where sexecode=2;
run;
```

Cela revient au même que de taper :

```
data malib.Table6;
set malib.Table;
if sexe='m' then delete;
run;
```

11. On peut séparer une table en deux comme ceci :

```
data malib.garcons malib.filles;
set malib.Table;
if sexecode=1 then output malib.garcons;
if sexecode=2 then output malib.filles;
run;
```

3 Graphiques et statistiques élémentaires

On s'intéresse à la série des pics d'ozone qui se trouve dans le jeu de données `ozone.xls` ; la variable `maxO3` correspond au maximum d'ozone pour chaque jour de la table.

1. Combien il y a-t-il d'observations ? Rajouter une variable `t` à ozone qui correspond au numéro de l'observation : utiliser une étape `data` et la commande `t=_n_`.

2. Tracer l'évolution au cours du temps de la variable `maxO3` à l'aide de la procédure `gplot` :

```
proc gplot data=malib.ozone;
plot maxO3*t;
run;
```

puis rajouter l'option : `symbol interpol=join;` Regarder dans l'aide les options de plot.

3. Tracer l'histogramme de la variable `maxO3` à l'aide de la procédure `gchart` :

```
proc gchart data=malib.ozone;
vbar maxO3;
run;
```

4. La procédure `means` permet d'obtenir quelques statistiques élémentaires :

```
proc means data=malib.ozone;
var maxO3;
run;
```

5. La procédure `univariate` permet d'obtenir un nombre plus important de statistiques. Utiliser-la pour obtenir la moyenne des observations `maxO3`, leur variance, coefficients d'aplatissement et d'asymétrie, médiane, distance interquartile, quantile à 5% et les valeurs extrêmes. En rajoutant l'option `plot` dans `proc univariate`, on obtient trois graphiques. Que représentent-ils ?

6. Même question pour la variable `T12` (température à midi).

7. Tracer le nuage de points d'abscisse `T12` et d'ordonnée `maxO3` avec `gplot`. Qu'est-ce que cela vous inspire ? Utiliser la `proc corr` pour préciser cela.