SAS

Travaux Dirigés

I. Création, modification et gestion des tables SAS

Références Bibliographiques

- *Introduction à SAS*, E. Duguet, Chapitres 1 à 5, éditions Economica.
- *SAS : Introduction au décisionnel*, S. Ringuedé, Chapitre 2, éditions Pearson.
- *A Handbook of Statistical Analyses with SAS,* G. Derr & B. S. Everitt, Chapitre 1, éditions CRC Press.

Pour toute question, commentaire ou suggestion, contacter : salim.lardjane@univ-ubs.fr

1) Saisir puis soumettre le programme simple suivant. Pour l'ensemble des raccourcis clavier, faire F9.

```
libname dir 'd:\data\sasdata';
data dir.bodyfat;
   input age pctfat 4.1 sexe$;
cards;
   23 279 H
   24 288 H
   27 278 H
   35 290 F
   38 290 H
   39 314 F
   41 25.9 F
   49 25.2 H
   50 31.1 H
   53 34.7 H
   53 42.0 F
   54 29.1 H
   56 32.5 H
   57 30.3 H
   58 330 F
   58 338 F
   60 411 F
   61 345 F
;
proc contents data = dir.bodyfat;
run;
proc print data = dir.bodyfat (firstobs=5 obs=10);
run;
```

On commence par définir la libref dir correspondant au répertoire d:\data\sasdata (spécifiez le répertoire de votre choix).

On utilise ensuite une étape **data** pour créer la table SAS bodyfat à sauvegarder dans la bibliothèque dir. Sour UNIX, l'extension du fichier obtenu est *ssd01*, sous SAS 6 Win, l'extension est *sd2*, sous SAS 7 Win et supérieures, l'extension est *sas7bdat* ou *sd7*.

L'instruction **cards** est utilisée pour saisir les données. Le point-virgule final doit être saisi après un retour à la ligne.

La **proc contents** permet d'afficher le contenu de la table SAS dir.bodyfat, c'est-à-dire les noms des variables qu'elle contient et leurs caractéristiques.

La **proc print** (avec les options **firstobs = 5** et **obs = 10**) permet d'afficher les observations 5 à 10 de la table dir.bodyfat.

2) Examiner le contenu des fenêtres Log et Output après soumission du programme.
3) Sauvegarder la table bodyfat sous forme de fichier texte à l'aide du programme suivant.

```
data _NULL_;
       set dir.bodyfat;
       file  'd:\data\sasdata\bodyfat.txt';
       put age sexe pctfat;
run;
```

Utiliser l'option **dlm=','** de **file** pour obtenir un fichier bodyfat.csv où les données sont séparées par des virgules.

4) Saisir la commande **fslist** dans la ligne de commande et ouvrir le fichier bodyfat.txt. Saisir **cols** dans la ligne de commande pour faire apparaître la réglette et **number** pour les numéros de lignes. L'ouvrir également en saisissant **notepad** dans la ligne de commande, puis **number** ; ensuite saisir **cols** en se positionnant sur un numéro de ligne. Notepad, contrairement à fslist permet d'effectuer des modifications sur le fichier. Les modifications sont sauvegardées lorsqu'on quitte notepad.

5) Lire les données du fichier bodyfat.txt dans une table SAS bodyfat2 à l'aide d'un *input de mode liste*. Le code correspondant est le suivant :

```
data bodyfat2;
       infile 'd:\data\sasdata\bodyfat.txt';
       input age sexe $ pctfat;
run;
proc print data = bodyfat2 (obs=5);
run;
```

6) On peut souhaiter mettre un fichier sous un format dit de transport (.tpt ou .xpt), permettant de passer d'une version de SAS à l'autre et notamment de SAS Windows à SAS Unix. Cela peut être fait à l'aide de la **proc cport**. Dans le cas de la table bodyfat, le code correspondant est le suivant.

```
proc cport data = bodyfat2 file = 'd:\data\sasdata\bodyfat2.tpt';
run;
```

7) Pour lire un fichier au format de transport, on utilise la procédure **cimport**. Par exemple, pour lire le fichier de transport précédemment créé, on peut utiliser le code suivant.

```
proc cimport data = bodyfat3 infile = 'd:\data\sasdata\bodyfat2.tpt';
run;
proc print data = bodyfat3 (firstobs = 5 obs = 10);
run;
```

8) Récupérer les fichiers nouvzel1.txt, nouvzel2.txt, nouvzel3.txt, nouvzel4.txt dans le répertoire …/duguet et les sauvegarder localement sur votre disque.

9) Examiner la structure des différents fichiers à l'aide de **fslist**.

10) On souhaite lire ces fichiers dans des tables SAS. Quels mode d'input vous semble adapté pour chaque fichier ? On rappelle qu'il existe trois modes principaux d'inputs sous SAS : liste, colonne et formaté. Répondre à cette question avant de passer aux suivantes. La liste des variables est donnée ci-dessous.

wbccode	Code du pays selon la banque mondiale
country	Nom du pays
year	année
pop	Population en milliers d'habitants
cgdp	Produit intérieur brut réel par habitant
y	Produit intérieur brut réel par habitant par rapport aux Etats-Unis (=100)
cc	Consommation réelle / produit intérieur brut réel (en %)
ci	Investissement réel / produit intérieur brut réel (en %)
cg	Dépenses publiques réelles / produit intérieur brut réel (en %)

11) Dans le fichier nouvzel1.txt, les données sont séparées par des espaces. On peut donc essayer un input de mode liste, par exemple à l'aide du code suivant :

```
data nouvzel1;
       infile 'd:\data\sasdata\nouvzel1.txt';
       input wbccode country $ year pop cgdp y cc ci cg;
run;
proc print data = nouvzel1;
run;
```

12) Le résultat obtenu est-il satisfaisant ? Pourquoi ? Que s'est-il passé ? Comment corriger le problème ?

13) On décide d'utiliser un input de mode colonne pour la variable country. Ceci peut être fait de la façon suivante.

```
data nouvzel1;
       infile 'd:\data\sasdata\nouvzel1.txt';
       input wbccode country $5-15 year pop cgdp y cc ci cg;
run;
proc print data = nouvzel1;
run;
```

14) Le résultat obtenu est-il satisfaisant ? Le code suivant conduirait-il au même résultat ?

```
data nouvzel1;
       infile 'd:\data\sasdata\nouvzel1.txt';
```

```
       input wbccode country $11. year pop cgdp y cc ci cg;
run;
proc print data = nouvzel1;
run;
```

De quel mode d'input s'agit-il ?

15) Quel mode d'input utiliser pour le fichier nouvzel2.txt ? On dispose de l'information suivante (dessin de fichier) :

wbccode	3.0 (longueur = 3)
country	$4-14 (longueur = 11)
year	4.0 (longueur = 4)
pop	4.0 (longueur = 4)
cgdp	4.1 (longueur = 4)
y	5.0 (longueur = 5)
cc	4.1 (longueur = 4)
ci	4.1 (longueur = 4)
cg	4.1 (longueur = 4)

16) On décide d'utiliser un input de mode formaté, le code correspondant s'écrivant :

```
data nouvzel2;
       infile 'd:\data\sasdata\nouvzel2.txt';
       input wbccode 3.0 country $11. year 4.0 pop 4.0 cgdp 4.1 y 5.0 cc 4.1
        ci 4.1 cg 4.1;
run;
proc print data = nouvzel2;
run;
```

17) Soumettre le code saisi. Le résultat obtenu est-il satisfaisant ?

18) Lire le fichier *nouvzel2.txt* à l'aide d'un input de mode colonne. A cet effet, on rappelle qu'il faut indiquer de façon précise la position des valeurs de chaque variable dans le fichier. Répondre à cette question avant de passer à la suivante.

19) Saisir et soumettre le code suivant :

```
data nouvzel2;
       infile 'd:\data\sasdata\nouvzel2.txt';
       input wbccode 1-3 country $4-14 year 15-18 pop 19-22
       cgdp 23-26 y 27-31 cc 32-35 ci 36-39 cg 40-43;
run;
proc print data = nouvzel2;
run;
```

20) Quelles sont les caractéristiques du fichier nouvzel3.txt ? Quel mode d'input utiliser ? Quelles difficultés vont se poser ?

21) Le fichier *nouvzel3.txt* se présente comme une seule ligne. Si on lance une instruction input simple dessus, par exemple le programme précédent, elle ne lira que les 43 premières colonnes de la ligne de données. Il faut donc empêcher SAS de passer automatiquement à la

ligne suivante après lecture des colonnes de données. Cela peut être fait en ajoutant @@ à la fin de l'instruction **input**. Par ailleurs, le logiciel ne lisant par défaut que 256 caractères par ligne, il faut lui indiquer la longueur totale de la ligne de données. On peut la déterminer dans la fenêtre **fslist** ou en examinant les propriétés du fichier *nouvzel3.txt* à l'aide de l'explorateur Windows. Ici, la longueur est de 1851 (octets). On la spécifie donc dans l'option **lrecl** (logical record length) de l'instruction **infile**. Le code obtenu est donc le suivant, pour un input de mode formaté :

```
data nouvzel3;
       infile 'd:\data\sasdata\nouvzel3.txt' lrecl=1851;
       input wbccode 3.0 country $11. year 4.0 pop 4.0
       cgdp 4.1 y 5.0 cc 4.1 ci 4.1 cg 4.1 @@;
run;
proc print data = nouvzel3;
run;
```

22) Quelle est la principale caractéristique du fichier nouvzel4.txt ? Pourquoi le dessin de fichier est-il inclus dans celui-ci ? Vous paraît-il intéressant de sauvegarder le code SAS de lecture du fichier dans le fichier même ? Pourquoi ?

23) L'option **firstobs =** de **infile** permet de spécifie la première ligne effective de données. Ce nombre peut être déterminé par exemple à l'aide de la fenêtre **fslist**. Le code obtenu dans ce cas est donc le suivant.

```
data nouvzel4;
       infile 'd:\data\sasdata\nouvzel4.txt' firstobs = 12;
       input wbccode country $11. year pop
       cgdp y cc ci cg;
run;
proc print data = nouvzel4;
run;
```

Il est également possible de lire le fichier jusqu'à une ligne donnée en utilisant l'option **obs =** de **infile**, suivie du numéro de la dernière ligne qu'on souhaite lire. Lire les 5 premières lignes de nouvzel4.txt dans une table SAS.

24) On aurait pu spécifier de façon plus concise les emplacements des différents fichiers lus de la façon suivante :

```
filename chemin 'd:\data\sasdata';
data nouvzel4;
       infile chemin(nouvzel4.txt) firstobs = 12;
       input wbccode country $11. year pop
       cgdp y cc ci cg;
run;
proc print data = nouvzel4;
run;
```

L'instruction **filename** permet de préciser un dossier dans lequel se trouvent les fichiers qu'on souhaite transformer en table SAS. Notons que **filename** peut également faire référence simplement à un fichier. Il s'agit, comme **libname**, d'une instruction globale.

25) Lancer R et sauvegarder les données iris en format délimité par des tabulations sous le nom iris.tab et délimité par des virgules sous le nom iris.csv. Cela peut être fait de façon adaptée à une lecture par SAS de la façon suivante.

```
> data(iris)

> write.table(iris,file="d:/data/sasdata/iris.csv",quote=FALSE,

sep=",",row.names=FALSE,col.names=TRUE)

> write.table(iris,file="d:/data/sasdata/iris.tab",quote=FALSE,

sep="\t",row.names=FALSE,col.names=TRUE)
```

26) Comment mettre ces données sous la forme de tables SAS à l'aide d'étapes data ? Quel mode d'input utiliser ? Répondre à ces questions avant de passer à la suivante.

27) Afin de lire le fichier iris.tab on utilise le code suivant.

```
data iris1;
       infile chemin(iris.tab) firstobs = 2 delimiter='09'x dsd pad;
       input Sepal_Length Sepal_Width Petal_Length Petal_Width Species $10.;
run;
proc print data = iris1;
run;
```

Que se passe-t-il lorsque certaines options de **infile** ne sont pas spécifiées ? Examiner les différents cas possibles.

28) Pour lire le fichier iris.csv, on utilise le code ci-dessous.

```
data iris2;
       infile chemin(iris.csv) firstobs = 2 dsd pad;
       input Sepal_Length Sepal_Width Petal_Length Petal_Width Species $10.;
run;
proc print data = iris2;
run;
```

Une alternative aurait pu être :

```
data iris2;
       infile chemin(iris.csv) firstobs = 2 dsd truncover;
       input Sepal_Length Sepal_Width Petal_Length Petal_Width Species $10.;
run;
proc print data = iris2;
run;
```

L'option **truncover** d'**infile**, utile lorsque la dernière modalité sur une ligne est de longueur variable, indique à SAS de ne pas rechercher sur la ligne suivante si on lui demande de lire un certain nombre de caractères pour une variable et qu'il en trouve moins. Examiner le résultat du programme précédent sans l'option **truncover**. Quelle est la différence de fonctionnement entre **truncover** et **pad** ?

29) Lire le fichier france.txt dans le répertoire …/duguet à l'aide d'une étape data. La liste des variables est donnée ci-dessous.

year	année
pop	Population en milliers d'habitants
y	Produit intérieur brut réel par habitant par rapport aux Etats-Unis (=100)
cgdp	Produit intérieur brut réel par habitant
cc	Consommation réelle / produit intérieur brut réel (en %)
ci	Investissement réel / produit intérieur brut réel (en %)
cg	Dépenses publiques réelles / produit intérieur brut réel (en %)

30) Le code utilisé est le suivant.

```
data france;
       infile chemin(france.txt);
       input year pop y cgdp cc ci cg;
run;
proc print data = france;
run;
```

31) On souhaitera dans un premier temps calculer le taux de croissance du produit intérieur brut (PIB) par habitant par année et, dans un deuxième temps, calculer une approximation de ce taux de croissance à l'aide du *logarithme* du PIB par habitant par année. Pour cela, nous n'avons besoin que d'une partie des données, plus précisément des variables year et cgdp. On souhaite donc construire une nouvelle table SAS à partir de la table france où seule les variables year et cgdp auront été conservées. Quelles instructions utiliser ?

32) Saisir puis soumettre le code suivant.

```
data france2;
       set france;
       keep year cgdp;
run;
```

33) Aurait-il été possible d'obtenir le même résultat avec l'instruction **drop** ? Donner le code correspondant. Pourquoi utilise-t-on plutôt **keep** dans la situation qui nous intéresse ?

34) On souhaite à présent modifier le nom des variables. Plus précisément, on souhaite renommer year par an et cgdp par pibr (PIB Réel). Cela peut être fait à l'aide de l'instruction **rename**. Le code correspondant est le suivant.

```
data france2;
       set france2;
       rename year = an cgdp = pibr;
run;
proc print data = france2;
run;
```

35) Créer, dans la table france2, une nouvelle variable lpibr correspondant au logarithme népérien du PIB réel. La fonction logarithme népérien est donnée sous SAS par la fonction log.

36) Saisir puis soumettre le code suivant.

```
data france2;
       set france2;
       lpibr = log(pibr);
run;
proc print data = france2;
run;
```

Obtient-on le résultat voulu ?

37) On souhaite à présent effectuer deux calculs : le calcul du taux de croissance exact du PIB, noté g, et le calcul d'une approximation de celui-ci par la différence des logarithmes. La formule exacte, si la variable est notée X_t, est la suivante

$$g_t = \frac{X_t - X_{t-1}}{X_{t-1}}$$

L'approximation considérée est donnée par

$$\hat{g}_t = \ln(X_t) - \ln(X_{t-1})$$

Afin de mener les calculs à bien, il nous faudra donc calculer des différences et utiliser des variables décalées dans le temps. Deux fonctions sont disponibles sous SAS pour ce faire : **lag** et **dif**. La fonction **lag** renvoie l'observation de la variable située sur la ligne précédente de la table SAS. La fonction **dif** effectue la différence entre l'observation courante de la variable et l'observation située sur la ligne précédente. Notons que la première valeur du calcul sera toujours manquante car elle correspond à la première observation. De plus, il est nécessaire, pour pouvoir utiliser ces fonctions, **que les données soient triées par ordre chronologique croissant**, sous peine d'obtenir un résultat qui n'a aucun sens. Cette condition est-elle vérifiée par nos données ? Quel code utiliser pour trier la table SAS par années croissantes ?

38) On utilise le code suivant

```
proc sort data = france2;
       by an;
run;
proc print data = france2;
run;
```

Notons que pour trier une table SAS selon deux variables ou plus, il suffit de les lister dans l'ordre souhaité dans l'instruction **by**. Pour effectuer un tri par valeurs décroissante, il faut spécifier l'option **descending** devant le nom de la variable de tri, dans l'instruction **by**.

39) Créer, à l'aide des fonctions **lag** et **dif**, deux nouvelles variables gpibr et dlpibr correspondant respectivement au taux de croissance exact et à son approximation par différence des logarithmes. Répondre à cette question avant de passer à la suivante.

40) Le code utilisé est le suivant.

```
data france2;
       set france2;
       gpibr = dif(pibr)/lag(pibr);
       dlpibr = dif(lpibr);
run;
```

```
proc print data = france2;
       id an;
       var pibr lpibr gpibr dlpibr;
run;
```

Les instructions **id** et **var** de la proc **print** permettent respectivement de spécifier une variable à utiliser comme identifiant des observations et la liste des variables à imprimer. Par ailleurs, aurait-il été possible de se passer de l'instruction **dif** ?

41) Comparer les taux de croissance obtenus par les deux méthodes en surimposant graphiquement les courbes correspondantes. Le code requis est le suivant (avec **sgplot**).

```
proc sgplot data = france2;
       xaxis  values = (1950 to 1992 by 3);
       series x = an y = gpibr;
       series x = an y = dlpibr;
run;
```

42) Définir une nouvelle variable erreur représentant la différence entre le taux de croissance exact et son approximation. Représenter graphiquement l'erreur commise en fonction du taux de croissance exact. L'approximation effectuée est-elle raisonnable ?

Le code requis est le suivant.

```
data france2;
       set france2;
       erreur = gpibr-dlpibr;
run;
proc sort data = france2;
       by gpibr;
run;
proc sgplot;
       series x = gpibr y = erreur;
run;
```

43) Récupérer la table SAS *frbe.sas7bdat* dans le répertoire …/duguet. On souhaite comparer les taux de croissance du PIB Réel entre la Belgique et la France, calculés sur la table frbe. Peut-on simplement réutiliser le programme vu précédemment ? Pourquoi ? Essayer de résoudre le problème avant de passer à la question suivante.

44) Afin de comparer l'évolution du taux de croissance du PIBR entre la Belgique et la France, on utilise le code suivant.

```
libname dir 'd:\data\sasdata';
data frbe;
       set dir.frbe;
run;
proc sort data = frbe;
       by country an;
run;
data frbe;
       set frbe;
       by country an;
       g=dif(pibr)/lag(pibr);
       if first.country then g=.;
run;
proc print data = frbe;
run;
```

A quoi peut correspondre l'instruction **first.country** ? Quel problème permet-elle de résoudre ?

45) L'instruction **first.country** génère une variable booléenne qui prend la valeur VRAI si l'on rencontre une modalité de country pour la première fois et prend la valeur FAUX sinon. Il faut, pour que cela fonctionne, que la table SAS soit triée au préalable selon la variable qui figure dans l'instruction **first** et que l'instruction **set** soit suivie d'une instruction **by** contenant cette même variable. Etait-il nécessaire de trier selon la variable an et de l'inclure dans l'instruction by ?

46) Comparer graphiquement l'évolution des taux de croissance du PIBR entre la Belgique et la France pour les données de la table frbe. Le code requis est le suivant.

```
proc sgplot data = frbe;
       xaxis values = ( 1966 to 1980 by 2);
       series x = an y = g / group = country;

run;
```

L'option **group** de l'instruction **series** permet d'obtenir une courbe par modalité de la variable spécifiée, ici country.

47) Récupérer les tables SAS fa et fb dans le répertoire …/duguet à l'aide du code suivant.

```
data fa;
       set dir.fa;
run;
data fb;
       set dir.fb;
run;
```

Visualiser les deux tables. On va s'intéresser à les combiner de diverses façons. On interprète la variable t comme l'année ou le numéro de période.

48) Est-il équivalent de combiner (concaténer) fa et fb à l'aide des deux programmes suivants ?

```
data tab;
       set fa fb;
run;
data tab;
       set fb fa;
run;
```

Utiliser des **proc print** pour répondre à la question.

Cette utilisation de l'instruction **set** est en fait très intéressante lorsqu'on possède des informations sur la même variable dans deux tableaux **différents**, par exemple les valeurs d'une même variable **pour deux ensembles de dates différentes**, dans le cadre des séries temporelles. Par contre dans l'exemple qui nous intéresse, on peut souhaiter obtenir une seule ligne pour chaque date alors que certaines dates sont présentes dans les deux tableaux. Cela peut être fait à l'aide de l'instruction **merge**, qui permet plus généralement de **fusionner** le contenu de plusieurs tableaux de diverses manières.

49) Afin d'effectuer une fusion de tableaux, on doit spécifier une variable-clef ; celle-ci correspond à un identifiant qui permet de repérer à quoi se rapporte l'information contenue dans les variables (par exemple : année, pays, individu…). Dans le cas des tables fa et fb, la variable-clef est la date t. Pour effectuer une **union** des deux tables, on peut utiliser le code suivant.

```
proc sort data = fa;
       by t;
run;
proc sort data = fb;
       by t;
run;
data tab;
       merge fa fb;
       by t;
run;
proc print;
       var x y z;
       id tabl t;
run;
```

Le tri de fa et fb selon t n'est en fait pas nécessaire ici car les deux tables étaient triées par date croissante à l'origine. Il est toutefois requis dans le cas général et il est donc préférable de **toujours** faire précéder un premier **merge** par les procédures de tri correspondantes.

50) Comme on peut le constater sur la variable tabl, si la même variable est présente dans les deux tableaux pour la même année, **seule la valeur lue dans la dernière table est conservée**. L'instruction merge ne traite donc pas les données de façon symétrique. L'ordre dans lequel les tables sont saisies est important. Par conséquent, **on doit toujours mettre les données**

d'une mise à jour dans la dernière table de l'instruction merge ; par exemple, on pourra mettre la table la plus récente en dernier.

51) L'instruction **merge** permet de réaliser des fusions plus compliquées que de simples unions. A cet effet, on utilise l'instruction **in =** qui permet de savoir si une modalité de la variable spécifiée par l'instruction by est présente dans un tableau ou non. Pour illustrer le fonctionnement de cette instruction, saisir le code suivant (les données sont supposées déjà triées par rapport à t).

```
data tab;
       merge fa (in = infa) fb (in = infb);
       by t;
       ta = infa;
       tb = infb;
run;
proc print;
       var x y z ta tb;
       id tabl t;
run;
```

Serait-il possible d'avoir simultanément, pour une même observation, ta = tb = 0 ?

52) Les variables ta et tb sont appelées variables booléennes, binaires ou dichotomiques. Sous SAS, ces différentes notions sont équivalentes : VRAI est codé par 1 et FAUX par 0. Lors de l'**union** de deux tables, les observations conservées sont celles pour lesquelles la proposition « ta OU tb » est VRAIE. Comment pourrait-on effectuer l'intersection des deux tables, c'est-à-dire ne conserver que les observations correspondant à des dates présentes dans les deux tableaux ?

53) Saisir puis soumettre le code suivant.

```
data tab;
       merge fa (in = a) fb (in = b);
       by t;
       if a and b;
run;
proc print;
       var x y z;
       id tabl t;
run;
```

En vous aidant des variables ta et tb définies précédemment, expliquer le fonctionnement de la fusion effectuée.

Comment obtenir une table contenant les observations correspondant à des dates présentes dans fa et pas dans fb, c'est-à-dire le complémentaire de fb par rapport à fa ?

54) On utilise le code suivant.

```
data tab;
       merge fa (in = a) fb (in =b);
       by t;
       if a and not(b);
run;
proc print;
```

```
       var x y z;
       id tabl t;
run;
```

Expliquer le fonctionnement de la fusion effectuée.

55) Comment utiliser les notions précédentes pour effectuer une **mise à jour** de la table fa à partir des informations contenues dans la table fb ? Répondre à cette question avant de passer à la suivante.

56) Le code utilisé est le suivant.

```
data tab;
       merge fa (in = a) fb;
       by t;
       if a;
run;
proc print;
       var x y z;
       id tabl t;
run;
```

Le résultat obtenu vous semble-t-il cohérent ?

Notons que les conditions que l'on met dans une étape data pour sélectionner les observations que l'on souhaite conserver sont parfois appelées **filtres**. Ainsi, dans les programmes précédents, o a utilisé successivement les filtres **a and b**, **a and not(b)** et **a**.

57) Pour pouvoir utiliser les différentes procédures de SAS sur un tableau de données, il est nécessaire que les variables soient rangées en colonnes et les observations en lignes. Il arrive toutefois que certains fichiers respectent la convention inverse. On doit alors **transposer** la table SAS obtenue en les lisant, ce qui peut être fait à l'aide de la procédure **transpose**. On illustrera le fonctionnement de cette procédure sur la table fa définie précédemment. A cet effet, saisir puis soumettre le code suivant.

```
proc transpose data = fa out = tab;
       var tabl x z;
run;
proc print;
run;
```

Les colonnes de la table SAS fa sont devenues les lignes de la table tab. La variable _NAME_ est automatiquement créée par SAS et contient simplement les noms des colonnes du tableau de départ. De même, des noms de variables sont automatiquement créés en rajoutant le numéro de ligne (dans le tableau de départ) au préfixe COL. Il est possible de modifier ce fonctionnement par défaut à l'aide de l'instruction **id** qui permet de spécifier le nom de la colonne du tableau de départ qui contient les noms des variables.

58) Saisir puis soumettre le code suivant.

```
proc transpose data = fa out = tab;
       var tabl x;
       id z;
run;
```

```
proc print;
run;
```

On aurait également pu utiliser une variable numérique, par exemple t ; toutefois, dans ce cas, SAS ajoute un caractère de soulignement devant le nombre afin d'obtenir un nom de variable admissible.

59) On peut en fait spécifier un préfixe à utiliser à la place du caractère de soulignement, si on le souhaite, à condition que la longueur des noms de variables obtenus ne soit pas supérieur à 8 caractère. Par exemple,

```
proc transpose data = fa out = tab prefix = annee;
       var tabl x z;
       id t;
run;
proc print;
run;
```

60) On souhaite à présent revenir au tableau de départ **à partir** du tableau tab. Ceci peut être fait à l'aide du code suivant.

```
proc transpose data = tab out = tab2 name = t;
       var annee1-annee10;
run;
proc print;
run;
data tab2;
       set tab2;
       t = substr(t,6,2);
       t0 = t-0;
       drop t;
       rename t0 = t;
run;
proc print;
run;
```

Essayer d'expliquer le fonctionnement du programme précédent en soumettant les instructions de façon graduelle et en s'aidant de la **proc contents** pour afficher la structure des tableaux obtenus.

61) On va s'intéresser à présent à différents aspects des opérations numériques, fonctions et boucles sous SAS. Commencer par saisir et soumettre le programme suivant.

```
data tab;
       do x = 0.01 to 1 by 0.01;
              y = log(x);
              output;
       end;
       label y ='ln(x)';
run;
proc print data = tab label;
       id x;
run;
```

Le compteur de la boucle, x, prend successivement les valeurs 0.01, 0.02, 0.03, …etc par pas de 0.01 jusqu'à la valeur 1. Utiliser les valeurs obtenues pour représenter graphiquement la

fonction logarithme népérien sur l'intervalle]0,1] à l'aide de **sgplot**. A quoi sert l'instruction **output** ?

62) On va s'intéresser à présent au calcul d'un taux d'inflation moyen. Si on note π_t le taux d'inflation entre les dates *t-1* et *t*, alors le taux d'inflation moyen sur la période allant de la date 1 à la date *T* est donné par la formule

$$\bar{\pi} = \left(\prod_{t=1}^{T} (1 + \pi_t) \right)^{1/T} - 1$$

Commencer par saisir et lire les données suivantes dans une table SAS appelée tab.

an	prixc
1989	3.4
1990	2.8
1991	3.2
1992	2.4
1993	2.2
1994	2.1
1995	1.6
1996	1.8
1997	1.1
1998	0.4
1999	0.5
2000	1.1

La variable prixc fournit le taux d'inflation en pourcentage ($100 \times \pi_t$).
Comment calculer le taux d'inflation moyen pour chacune des périodes 1989-1989, 1989-1990, …, 1989-2000 à l'aide d'un programme SAS ? Essayer de concevoir un tel programme avant de passer à la question suivante.

63) On peut utiliser le programme suivant.

```
data tab;
       set tab;
       m = 1 + prixc/100;
       retain cumul;
       if _n_=1 then cumul = m;
       else cumul = cumul * m;
       tm = (cumul**(1/_n_)-1)*100;
run;
proc print;
       id an;
run;
```

Expliquer le fonctionnement du programme. Quelle est la fonction de l'instruction **retain** ? Aurait-on pu effectuer le calcul en faisant intervenir l'instruction **lag**, vue précédemment ?

64) On s'intéresse à présent à la détermination de l'évolution du capital d'un épargnant au cours du temps, sachant que le taux d'intérêt et le montant épargné varient chaque année. On note r_t le taux d'intérêt annuel qui s'applique aux fonds déposés en début d'année *t* et S_{t-1} l'épargne supplémentaire investie au début de l'année *t*, qui vient s'ajouter au capital de

l'année précédente. A la fin de la première année, l'épargnant a sur son compte le capital K_1 donné par

$$K_1 = (1 + r_1)S_0$$

A partir de la deuxième année et à la fin de chaque année, l'épargnant touche les intérêts du capital de l'année précédente, qu'on suppose entièrement réinvestis, et les intérêts sur l'épargne supplémentaire S_{t-1} qu'il a placée en début d'année, i.e.

$$K_t = (1 + r_t)(K_{t-1} + S_{t-1}), \quad t \geq 2$$

Les taux d'intérêt et les montants épargnés sont les suivants.

t	interet	epargne
1	0.10000	10
2	0.10000	20
3	0.07492	30
4	0.10000	40
5	0.10000	50
6	0.07014	60
7	0.00509	70
8	0.01534	80
9	0.02360	90
10	0.10000	100
11	0.09696	110
12	0.03716	120
13	0.10000	130
14	0.05018	140
15	0.07934	150
16	0.10000	160
17	0.10000	170
18	0.07386	180
19	0.04550	190
20	0.04875	200

Lire les données précédentes dans une table SAS appelée tab.
Ecrire un programme SAS permettant de calculer l'évolution du capital de l'épargnant au cours du temps. Répondre à cette question avant de passer à la suivante.

65) On peut utiliser le code suivant.

```
data tab;
       set tab;
       retain capital;
       if t = 1 then capital = (1+interet)*epargne;
       else capital = (1+interet)*(capital+epargne);
       label t = 'Durée du placement';
run;
proc print;
       id t;
run;
```

Représenter sur un même graphique l'évolution du capital et du taux d'intérêt à l'aide la procédure **sgplot**. On pourra utiliser l'instruction **xaxis** et l'option **y2axis** pour améliorer la présentation du graphique.

66) On va s'intéresser à présent à la manipulation de chaînes de caractères sous SAS. Les opérations usuelles sont données dans le tableau ci-dessous.

Fonction	*Définition*
compress(x)	Supprime les espaces dans la chaîne x
compress(x,'a')	Supprime les caractères a de la chaîne x
left(x)	Aligne la chaîne x à gauche
right(x)	Aligne la chaîne x à droite
substr(x,a,b)	Extrait une chaîne de x, de longueur b à partir de la position a
length(x)	Longueur de la chaîne x
x||y	Concatène les chaîne x et y
length x $a	Déclare une chaîne x de longueur a

Afin d'illustrer le fonctionnement de ces différentes fonctions, saisir et lire dans une table SAS appelée tab les données suivantes

country
austria
belgium
cyprus
denmark
france
germany
greece
hungary
italy
turkey
utd kingdm

puis soumettre le programme suivant :

```
data tab;
	set tab;
	a = compress(country);
	b = compress(country,'e');
	c = left(country);
	d = right(country);
	e = substr(country,2,3);
	f = length(country);
	label
		a = 'compress(country)'
		b = 'compress(country,"e")'
		c = 'left(country)'
		d = 'right(country)'
		e = 'substr(country,2,3)'
		f = 'length(country)'
	;
run;
proc print label;
```

```
        id country;
run;
```

Essayez de deviner quelles seront les sorties obtenues puis examinez-les.

67) On va s'intéresser à présent plus spécifiquement à la **concaténation** de chaînes de caractères.

Saisir et lire dans une table SAS appelée tab les données suivantes.

i	*x*	*y*
1	X1	Y1
2	X2	Y2
3	X3	Y3
4	X4	Y4
5	X5	Y5
6	X6	Y6
7	X7	Y7
8	X8	Y8
9	X9	Y9
10	X10	Y10
11	X11	Y11
12	X12	Y12
13	X13	Y13
14	X14	Y14
15	X15	Y15
16	X16	Y16
17	X17	Y17
18	X18	Y18
19	X19	Y19
20	X20	Y20

68) Saisir puis soumettre le programme suivant.

```
data tab;
        set tab;
        z1 = compress(x||y);
        z2 = x||y;
        label
                z1 = 'compress(x||y)'
                z2 = 'x||y'
        ;
run;
proc print label;
        var z1 z2;
        id i;
run;
proc contents;
run;
```

Quelle est l'utilité de **compress(x||y)** par rapport à **x||y** ?

69) On peut également spécifier un symbole à insérer à l'aide de l'opérateur de concaténation. Par exemple,

```
data tab;
       set tab;
       z = compress(x||'+'||y);
       label z = 'compress(x||"+"||y)';
run;
proc contents;
run;
proc print label;
       var z;
       id i;
run;
```

70) On peut parfois avoir besoin de concaténer des chaînes de caractères de façon incrémentale. Dans ce cas-là, on doit utiliser l'instruction **retain** vue précédemment. Par exemple,

```
data tab;
       set tab;
       length cumul $60;
       retain cumul;
       if _n_=1 then cumul = x;
       else cumul = compress(cumul||x);
run;
proc print;
       var cumul;
       id i;
run;
```

Que se passe-t-il si on omet de fixer la longueur de cumul à 60 ?