SAS®
Initiation et manipulation de données
Combiner des tables SAS

Nicolas Poulin

poulin@math.unistra.fr

Combiner des tables SAS

- Il s'agit de créer une table SAS à partir de plusieurs tables SAS.

- Il y a 2 types de combinaisons :

  - Mettre dans une même table SAS des observations provenant de plusieurs tables SAS.

  - Mettre dans une même table SAS des variables provenant de plusieurs tables SAS.

poulin@math.unistra.fr

Ajouter et concaténer

- Il s'agit de mettre dans une même table SAS des observations provenant de plusieurs tables SAS. Dans la table SAS obtenue, il y aura les lignes des tables SAS d'origine.

- SAS propose 2 méthodes :

  - Ajouter (appending) : ajoute les observations de la deuxième table directement à la fin de la première table.

  - Concaténation : génère une copie de toutes les observations de la première table puis copie toutes les observations d'une ou plusieurs tables dans une nouvelle table.

poulin@math.unistra.fr

Appending

- La procédure APPEND permet d'ajouter les observations d'une table SAS à la fin d'une autre table SAS.

```
PROC APPEND BASE=libref.filename1
            DATA=libref.filename2;
RUN;
```

- BASE : nom de la table à laquelle les observations sont ajoutées.
- DATA: nom de la table dont les observations seront ajoutées à celle de la table définie en BASE

poulin@math.unistra.fr

Appending

- Seulement deux tables SAS peuvent être utilisées en même temps.
- Les observations contenues dans la table BASE ne sont pas lues (seules celles de la table DATA seront copiées dans le PDV).
- Le bloc descripteur de la table BASE ne peut pas être modifié.

poulin@math.unistra.fr

Créer un jeu de données

```
DATA table1;
        INPUT nom $ age sexe $;
        DATALINES;
        Chris 36 M
        Jane 21 F
        Tom 30 M
        ;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table2;
        INPUT nom $ age sexe $;
        DATALINES;
        Alice 27 F
        Bob 25 M
        Harry 30 M
        ;
RUN;
```


poulin@math.unistra.fr

Appending : exemple 1

```
PROC APPEND BASE=table1
            DATA=table2;
RUN;

PROC PRINT DATA=table1;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table2b;
        LENGTH nom $6. sexe $1.;
        INPUT nom $ age sexe $;
        DATALINES;
        Alice 27 F
        Bob 25 M
        Harry 30 M
        ;
RUN;
```


poulin@math.unistra.fr

Appending : exemple 2

```
PROC APPEND BASE=table1
            DATA=table2;
RUN;
```

```
PROC CONTENTS DATA=table1;
RUN;
```

```
PROC PRINT DATA=table1;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table3;
        INPUT nom $ age ;
        DATALINES;
        Joe 49
        Jerry 28
        ;
RUN;
```


poulin@math.unistra.fr

Appending : exemple 3

```
PROC APPEND BASE=table1
            DATA=table3;
RUN;

PROC PRINT DATA=table1;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table4;
        INPUT nom $ age Pays $ ;
        DATALINES;
        Tim 39 UK
        Helen 23 USA
        ;
RUN;
```


poulin@math.unistra.fr

Appending : exemple 3

```
PROC APPEND BASE=table1
            DATA=table4;
RUN;

PROC PRINT DATA=table1;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table2c;
        INPUT nom $ age $ sexe $;
        DATALINES;
        Alice 27 F
        Bob 25 M
        Harry 30 M
        ;
RUN;


PROC CONTENTS DATA=table2c;

RUN;
```


poulin@math.unistra.fr

Appending : exemple 4

```
PROC APPEND BASE=table1
            DATA=table2c;
RUN;
```


poulin@math.unistra.fr

Appending : exemple 4

```
PROC APPEND BASE=table1
            DATA=table2c;
RUN;
```

Remarque

Ce code produit une erreur car la variable age est définie en tant que numérique dans table1 et en tant que texte dans table2c.
On va forcer SAS à passer outre cette erreur en utilisant l'option FORCE.

poulin@math.unistra.fr

Appending : exemple 4

```
PROC APPEND BASE=table1
            DATA=table2c FORCE;
RUN;


PROC CONTENTS DATA=table1 ;
RUN;


PROC PRINT DATA=table1 ;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table2d;
        INPUT nom $ age $ sexe $;
        DATALINES;
        Alice 27y F
        Bob 25y M
        Harry 30y M
        ;
RUN;


PROC CONTENTS DATA=table2c;

RUN;
```


poulin@math.unistra.fr

Appending : exemple 5

```
PROC APPEND BASE=table1
            DATA=table2d FORCE;
RUN;


PROC CONTENTS DATA=table1 ;
RUN;


PROC PRINT DATA=table1 ;
RUN;
```


poulin@math.unistra.fr

Concaténer

- La procédure APPEND ne permet que de copier les observaitons d'une seule table à une table d'entrée.
- La concaténation permet de mettre à la suite des observations issues de plusieurs tables SAS pour en créer une nouvelle.
- La concaténation se fait via une étape DATA.

poulin@math.unistra.fr

Concaténer

```
DATA BASE=filename1
        SET filename2 ··· filenameN;
RUN;
```

- L'instruction SET peut contenir autant de tables que souhaité.
- Les observations seront stockées dans l'ordre des tables déclarées dans l'instruction SET.
- Toutes les observations seront lues (PDV).

poulin@math.unistra.fr

Concaténer : exemple 1

Remarque

La table1 utilisée dans les exemples suivants est la table1 initialement utilisée pour les exemples avec la PROC APPEND.

```
DATA tablec;

        SET table1 table2;

RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 2

```
DATA tablec;
        SET table1 table2b;
RUN;
```

```
PROC CONTENTS DATA=tablec ;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 2

```
DATA tablec;
        SET table2b table1;
RUN;



PROC CONTENTS DATA=tablec ;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 2

```
DATA tablec;
        SET table2b table1;
RUN;
```

```
PROC CONTENTS DATA=tablec ;
RUN;
```

Remarque

La première table déclarée dans l'instruction SET est utilisée pour créer le bloc descripteur de la table de sortie.
Utiliser l'instruction LENGTH si nécessaire.

poulin@math.unistra.fr

Concaténer : exemple 3

```
DATA tablec;
        SET table1 table2c;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 4

```
DATA tablec;
        SET table1 table2 table3;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 5

```
DATA tablec;
        SET table1 table2 table4;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table2e;
        INPUT nom $ age genre $;
        DATALINES;
        Alice 27 F
        Bob 25 M
        Harry 30 M
        ;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 6

```
DATA tablec;

        SET table1 table2e;

RUN;
```

Remarque

Les variables sexe et genre sont en fait une seule et même variable mais avec 2 noms différents.

poulin@math.unistra.fr

Concaténer : exemple 6

On peut forcer SAS à renommer une variable.

```
DATA tablec;
        SET table1(rename=(sexe=genre)) table2e;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 7

On peut intercaler des observations des 2 tables en demandant que les lignes soient triées en fonction d'une variable.

```
DATA tablec;

    SET table1 table2;

    BY nom;

RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table5;
        INPUT nom $ age sexe $;
        DATALINES;
        Yann 42 M
        Franck 29 M
        Julie 41 F
        ;
RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 8

```
DATA tablec;

        SET table1 table5;

        BY nom;

RUN;
```


poulin@math.unistra.fr

Concaténer : exemple 8

```
DATA tablec;
        SET table1 table5;
        BY nom;
RUN;
```

Remarque

Comme on utilise un BY il faut que toutes les tables soient triées.

poulin@math.unistra.fr

Concaténer : exemple 8

```
PROC SORT DATA=table5;
          BY nom;
RUN;

DATA tablec;
     SET table1 table5;
     BY nom;
RUN;
```


poulin@math.unistra.fr

Doublons

- On peut demander à SAS de supprimer les doublons grâce à la PROC SORT.

```
DATA tablec;
        SET table1 table2 table3 table4 table5;
        BY nom;
RUN;
```

- L'instruction NODUPKEY permet de ne garder qu'une seule ligne par valeur de la variable utilisée dans le BY.

```
PROC SORT DATA=tablec NODUPKEY;
                BY nom;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table5b;
        INPUT nom $ age sexe $;
        DATALINES;
        Yann 27 M
        Franck 29 M
        Julie 41 F
        ;
RUN;
```


poulin@math.unistra.fr

Doublons

```
DATA tablec;
        SET table1 table5 table5b;
RUN;

PROC SORT DATA=tablec NODUPKEY;
                BY nom;
RUN;
```


poulin@math.unistra.fr

Doublons

```
DATA tablec;
        SET table1 table5 table5b;
RUN;
```

- L'instruction NODUP permet de supprimer les doublons (mêmes valeurs pour chaque variable).

```
PROC SORT DATA=tablec NODUP;
                BY nom;
RUN;
```


poulin@math.unistra.fr

Fusion (Merging)

- La fusion consiste à combiner les observations de plusieurs tables SAS en une seule dans une nouvelle table SAS.
- Afin de fusionner plusieurs tables il faut que celles-ci aient une variable commune qui soit un marqueur unique de l'individu.
- Cette méthode de fusion s'appelle fusion par concordance (match-merging).
- Il est possible d'utiliser plusieurs variables pour la concordance.
- Les tables initiales doivent être triées en utilisant la (les) variable(s) de concordance.

poulin@math.unistra.fr

Créer un jeu de données

```
DATA table1;
        INPUT prenom $ age sexe $ id;
        DATALINES;
        Chris 36 M 1
        Jane 21 F 2
        Tom 30 M 3
        ;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA table2;
        INPUT nom $ id;
        DATALINES;
        Alice 1
        Bob 3
        Harry 2
        ;
RUN;
```


poulin@math.unistra.fr

Match-merging

```
DATA filename1;
        MERGE filename2 ··· filenameN;
        BY var1 ··· varN;
RUN;
```

- L'instruction SET peut contenir autant de tables que souhaité.
- L'instruction BY peut contenir autant de tables que souhaité.

poulin@math.unistra.fr

Match-merging : exemple

```
PROC SORT DATA=table1 ;
          BY id;
RUN;

PROC SORT DATA=table2 ;
          BY id;
RUN;

DATA tablef;
     MERGE table1 table2;
     BY id;
RUN;
```


poulin@math.unistra.fr

Match-merging : exemple

- On peut utiliser l'instruction RETAIN pour ordonner les variables dans la table créée.

```
DATA tablef;
        RETAIN id nom prenom;
        MERGE table1 table2;
        BY id;
RUN;
```


poulin@math.unistra.fr

Concordance de 1 à plusieurs

- Jusqu'ici on a pris un exemple d'une concordance de 1 à 1 : 1 ligne dans la première table concorde avec 1 ligne dans la seconde table.
- Il est possible d'avoir une concordance de 1 à plusieurs : 1 ligne dans la première table concorde avec plusieurs lignes dans la seconde table.
- La ligne de concordance dans la première table sera répétée plusieurs fois pour être mise en concordance avec les lignes de la seconde table.

poulin@math.unistra.fr

Créer un jeu de données

```
DATA phone;
     INPUT type $ id tel;
     DATALINES;
     perso 1 0606060606
     perso 3 0680808080
     perso 2 0655555555
     pro 1 0706060606
     pro 3 0780808080
     pro 2 0755555555
     ;
RUN;
```


poulin@math.unistra.fr

Match-merging : exemple

```
PROC SORT DATA=phone ;
          BY id;
RUN;
```

```
DATA tablef;
     MERGE table1 phone;
     BY id;
RUN;
```


poulin@math.unistra.fr

Créer un jeu de données

```
DATA phone2;
     INPUT type $ id tel;
     DATALINES;
     perso 1 0606060606
     perso 4 0680808081
     perso 3 0680808080
     pro 1 0706060606
     pro 3 0780808080
     pro 4 0755555555
     ;
RUN;
```


poulin@math.unistra.fr

Match-merging : exemple

```
PROC SORT DATA=phone2;
          BY id;
RUN;

DATA tablephone2;
     MERGE table1 phone2;
     BY id;
RUN;
```


poulin@math.unistra.fr

Option de table IN=

- Comme nous l'avons vu dans l'exemple précédent, si une valeur de la variable de concordance n'est présente que dans une table, la table créée comportera les lignes pour l'ensemble des valeurs (sur les 2 tables) de la variable de concordance.
- Cela peut générer pas mal de données manquantes.
- Il est possible de générer une table ne comportant que les observations pour lesquelles la valeur de la variable de concordance est présente dans les 2 tables.
- Il est aussi possible de générer une table ne comportant que les observations pour lesquelles la valeur de la variable de concordance n'est présente que dans une des 2 tables.

poulin@math.unistra.fr

Option de table IN=

- L'option de table IN= crée une variable booléenne qui indique si la variable de concordance de cette table fait partie de l'observation en cours.

- La syntaxe générale est la suivante.
  filename(IN=variable)

- Une variable temporaire (dans le PDV) sera créée et aura 2 valeurs possibles.

0	Indique que l'observation courante ne contient pas la valeur de concordance.
1	Indique que l'observation courante contient la valeur de concordance.

poulin@math.unistra.fr

IN = : exemple

- Il faut rajouter une instruction IF pour sélectionnes les lignes à considérer.

```
DATA tablephone2;

    MERGE table1(in=test1) phone2(in=test2);

    BY id;

    IF test1=1 AND test2=1;

RUN;
```


poulin@math.unistra.fr

IN = : exemple

```
DATA tablephone2;
        MERGE table1(in=test1) phone2(in=test2);
        BY id;
        IF test1=0 AND test2=1;
RUN;
```


poulin@math.unistra.fr

IN = : exemple

```
DATA tablephone2;
        MERGE table1(in=test1) phone2(in=test2);
        BY id;
        IF test1=1 AND test2=0;
RUN;
```


poulin@math.unistra.fr