



Manipulation des données sous SAS (2)

Clément TASSART
Agnès BAUMER
Sanja SAMARDZIC
Sabrina MECHETTA



Vue d'ensemble

- Les datasets
- Opérations sur les datasets
- Les fusions / jointures
- Utilisation du SQL
- Les macro-programmes
- Exportation des données


Les datasets

Les vecteurs

- Pour créer des vecteurs, ou des matrices, on va faire appel à la **PROC IML**

Pour créer simplement un vecteur


```
19 proc iml;  
20 x1={1 2 3 4 5};  
21 print x1;  
22 run;
```



x1				
1	2	3	4	5

Pour incrémenter de 1


```
24 proc iml;  
25 x2=(1:5);  
26 print x2;  
27 run;
```



x2				
1	2	3	4	5

Pour incrémenter d'une valeur y

```
29 proc iml;  
30 x3=do(1,3,0.5);  
31 print x3;  
32 run;
```



x3				
1	1.5	2	2.5	3

Les matrices

Pour créer simplement une matrice

```
19 proc iml;  
20 x={1 2 3, 4 5 6};  
21 print x;  
22 run;
```

x		
1	2	3
4	5	6

$nrow(x) = 2$

$ncol(x) = 3$

Pour créer un matrice constante

```
24 proc iml;  
25 y=j(2,2,2);  
26 print y;  
27 run;
```

$y[1,1] = 2$

y	
2	2
2	2

Les matrices

On peut également ajouter des noms de colonnes et de lignes et effectuer des opérations

```
47 proc iml;
48 Origine = {"Lyon 2" "Exté"};
49 Sexe = {"F" "H"};
50 L = "M2 SISE";
51 Classe = {4 5, 8 6};
52 Id=j(2,1,1);
53 Classe2= Classe*Id;
54 print Classe[rowname=Sexe colname=Origine label=L],
55 Classe2[rowname=Sexe colname="nombre" label=L];
```

M2 SISE		
	Lyon 2	Exté
F	4	5
H	8	6

M2 SISE	
	nombre
F	9
H	14

multiplication matricielle

Les datasets

- créer un dataset "à la main"

```

1 DATA df;
2 INPUT Matiere$ coeff ;
3 DATALINES;
4 Mathematiques 15
5 Francais 12
6 Anglais 5
7 ;
8 RUN;

```

\$ pour spécifier que c'est une **chaîne de caractère** (on peut également spécifier le nombre de caractères par exemple \$2. pour 2 caractères)

	Matiere	coeff
1	Mathemat	15
2	Francais	12
3	Anglais	5

- importer un dataset existant

```

10 proc import datafile = "coefficients_matières.xls"
11      out = coeff
12      dbms= xls
13 format replace;
14
15      getnames=yes;
16      sheet= 'annee 2019'

```

fichier à importer

nom de la table de sortie

obtenir le nom des variables

nom de la feuille excel à sélectionner

Les datasets

On peut aussi créer un dataset à partir d'une matrice

```
47 proc iml;
48 Origine = {"Lyon 2" "Exté"};
49 Sexe = {"F" "H"};
50 L = "M2 SISE";
51 Classe = {4 5, 8 6};
52 Id=j(2,1,1);
53 Classe2= Classe*Id;
54 create Table from Classe[rowname=Sexe colname=Origine];
55 append from Classe[rowname=Sexe];
56 run;
```

nom du dataset
qu'on crée

nom de la matrice d'origine



bien préciser rowname dans cette
ligne, sinon il n'est pas affiché

Opérations sur les datasets

Insertions et suppressions

- On peut ajouter une nouvelle ligne en utilisant **data input** fonction et après **data set** fonction
- On supprime les lignes avec la fonction **delete** mais il faut spécifier avec **where** ou **if** statement quelle ligne nous voulons supprimer. Si non, toutes les lignes disparaîtront

```

26 /* Ajouter une nouvelle ligne et afficher le resultat */
27
28 data new_data;
29   infile datalines delimiter=",";
30   input sep_length sep_width pet_length pet_width species;
31   datalines;
32 5.5,2.6,6.9,.,3
33 ;
34 run;
35
36 data practice.iris;
37 set practice.iris work.new_data;
38 run;
39
40 /* Supprimer cette ligne et afficher le resultat */
41 data practice.iris;
42   set practice.iris;
43   if missing(pet_width) then delete;
44 run;

```

- On peut facilement créer une nouvelle colonne en écrivant une fonction (sum, max, min, mean)

```

1 data eu_occ_;
2   set library_name.eu_occ;
3   somme=sum(hotel,shortstay,camp)
4 run;
5

```

Filtres et tris

```
58 /* Filtrage des donnees en utilisant WHERE statement */
59 data subset;
60     set work.subset_iris;
61     where pet_length>3 and pet_width<1.5;
62 run;
63
64 data subset;
65     set work.subset_iris;
66     where pet_width is missing;
67     keep pet_width pet_length species;
68 run;
69
70 /* Trier les donnees */
71 proc sort data=subset_iris out=sorted;
72     by descending sep_width;
73 run;
74
75 /* Trouver les valeurs duplicates et les mettre dans une nouvelle table*/
76 proc sort data=subset_iris out=sorted
77     nodupkey dupout=duplicate_iris;
78     by _all_;
79 run;
```

← filtrage des valeurs manquantes

← ascending by default

← Il faut spécifier “dupout” table pour situer toutes les valeurs duplicates

Les conditions

```

81 /* Conditional processing */
82 /* IF-THEN-ELSE */
83 data subset_iris;
84     set practice.iris;
85     length latin_name $ 10;
86     if species=1 then latin_name='setosa';
87     else if species=2 then latin_name='versicolor';
88     else if species=3 then latin_name='virginica';
89 run;

```

Si on veut définir plusieurs déclarations selon une condition (par. ex. si species=1), on doit utiliser IF-THEN-DO fonction. Il ne faut pas oublier à fermer chaque “do” avec “end”.

Au lieu de cette fonction, on peut utiliser SELECT-WHEN fonction

```

100 /* Conditional processing */
101 /* IF-THEN-DO */
102 data subset_iris;
103     set practice.iris;
104     length latin_name $ 10;
105     length origin $ 18;
106     if species=1 then do;
107         latin_name='setosa';
108         origin="Japanese";
109     end;
110     else if species=2 then do;
111         latin_name='versicolor';
112         origin="America and Canada";
113     end;
114     else if species=3 then do;
115         latin_name='virginica';
116         origin="North America";
117     end;
118 run;

```

Les fusions/jointures (MERGE)

Principe

Obs.	sepal_length	sepal_width	petal_length	petal_width	variety
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3	1.4	0.2	Setosa
3	6.2	2.9	4.3	1.3	Versicolor
4	5.1	2.5	3	1.1	Versicolor
5	5.7	2.8	4.1	1.3	Versicolor
6	6.2	3.4	5.4	2.3	Virginica
7	5.9	3	5.1	1.8	Virginica

dataset iris

Obs.	variety	binomial_name
1	Setosa	Iris setosa
2	Versicolor	Iris versicolor
3	Virginica	Iris virginica

dataset iris_names

Obs.	sepal_length	sepal_width	petal_length	petal_width	variety	binomial_name
1	5.1	3.5	1.4	0.2	Setosa	Iris setosa
2	4.9	3	1.4	0.2	Setosa	Iris setosa
3	6.2	2.9	4.3	1.3	Versicolor	Iris versicolor
4	5.1	2.5	3	1.1	Versicolor	Iris versicolor
5	5.7	2.8	4.1	1.3	Versicolor	Iris versicolor
6	6.2	3.4	5.4	2.3	Virginica	Iris virginica
7	5.9	3	5.1	1.8	Virginica	Iris virginica

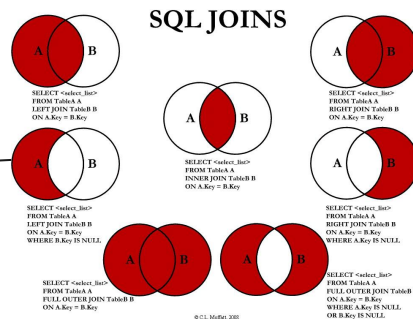
dataset iris_final

Prérequis

- Préparation des données (prérequis)
 - Deux jeux de données ayant au moins une **variable commune**
 - Là où les variables communes doivent avoir le **même nom**
 - Ils doivent être **triés** sur cette/ces variable(s)

Avantages de MERGE

- ✓ Toutes les jointures sont possibles comme en SQL
- ✓ Les jointures avec des clés dupliquées fonctionnent



Syntaxe

- Tris des jeux de données sur la variable commune (Procédure **SORT**)

- Jointure des deux jeux de données (Statement **MERGE**)

- iris_final représente le nouveau jeu de données issu de la jointure.

```
1 * Import du jeu de données iris ;
2 * ...
3
4 * Tri du jeu de données iris sur la variable variety ;
5 PROC SORT DATA=iris;
6     BY variety;
7 RUN;
8
9 * Import du jeu de données iris_names ;
10 * ...
11
12 * Tri du jeu de données iris_names sur la variable variety ;
13 PROC SORT DATA=iris_names;
14     BY variety;
15 RUN;
16
17 * Jointure des deux jeux de données en un seul (iris_final) par la variable variety ;
18 DATA iris_final;
19     MERGE iris iris_names;
20     BY variety;
21 RUN;
```

Variable commune

Ressources



- <https://kb.iu.edu/d/afn>
- https://www.sas.com/content/dam/SAS/en_ca/User%20Group%20Presentations/Montreal-User-Group/Guerss-MergeVsJoin.pdf (pages 3 à 6)
- <https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/178-2008.pdf> (page 5)

Utilisation du SQL

Structured Query Language

Pourquoi SQL ?

→ Alternative à l'étape data

- Extraire, corriger, mettre à jour des données dans une table SAS
- Créer une vue logique ou une table
- Lister le contenu d'une table avec des restrictions sur certaines variables et/ou observations
- Fusionner et/ou joindre des tables
- Trier une table
- Synthétiser

Avantages

- ✓ Souvent plus rapide que les étapes data
- ✓ Commun à de nombreux logiciels de gestion de bases de données (SGBD)
- ✓ Langage simplifié

Syntaxe

- Entre `proc sql` et `quit` : uniquement du SQL
- `noprint` : permet de ne pas afficher les résultats
- Utilisation du langage SQL "classique" : condition, union, jointures, produit cartésien...

RESULTATS

Species	Somme	Moyenne
setosa	73.1	5.006
virginica	277.6	6.588

```

21 proc sql noprint;
22
23 * Création d'une table ;
24 create table iris_virginica_setosa
25 as select *
26 from iris
27 where species like "virginica" or species like "setosa" ;
28
29 * Jointure ;
30 create table iris_code as
31 select a.*, b.code
32 from iris_virginica_setosa as a left join
33 (select unique(Species), substr(Species, 1, 3) as code
34 from Iris) as b
35 on a.Species=b.Species;
36
37 * Modifications;
38 insert into iris_code
39 values("Autres",5,5,5,5,"au");
40 update iris_code
41 set Species=upper(Species);
42 delete from iris_code where Petal_length+Sepal_length<10;
43
44 * Création d'une vue ;
45 select Species,sum(Petal_Length) as Somme,mean(Sepal_Length) as Moyenne
46 from iris_virginica_setosa
47 group by Species;
48
49 quit;

```

DONNEES EN SORTIE

Lignes totales : 100 Colonne(s) totales : 5

	Species	Sepal_length	Sepal_Width
1	setosa	5.1	3.5
2	setosa	4.9	3
3	setosa	4.7	3.2
4	setosa	4.6	3.1

Utilisation des fonctions SAS

Ressources



- http://maths.cnam.fr/IMG/pdf/SAS_COURS_1-2.pdf section 4.7 (page 53)
- <https://sql.sh> (cours de SQL)
- <http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002294523.htm> (documentation officielle de la procédure proc SQL SAS)

Les macros (programmes) SAS

Pourquoi des macros ?

- Améliore les possibilités du langage de base
 - Passage de paramètres entre les étapes DATA et PROC
 - Augmente la rapidité des écritures
 - Automatise les programmes
 - Facilite l'utilisation

Macro-variable

- Pas rattachée à une table SAS
- Contient une seule valeur
- Systématiquement du texte

Macros

- S'applique à la valeur d'une macro variable stockée en mémoire pendant la session
- Utilisée n'importe quand

Variable SAS

- Rattachée à une table SAS
- Une valeur par observation de la table SAS
- Type numérique ou caractère

Fonctions

- S'applique à la valeur d'une variable stockée dans une table
- Utilisée dans les étapes data et, parfois, les procédures

Macro-variable ≠ Variable SAS

Macros ≠ Fonctions

Les macros-variables

Syntaxe de base :

- Déclaration : `%let nomVar=valeur` (chaîne de caractère);
- Affichage : `%put &nomVar;`
- Référence : `&nomVar;`
- Suppression : `%symdel nomVar;`

Initialisations :

- `%global nomVar_globale` ; variable globale
- `%local nomVar_locale` ; variable locale

Liens avec les tables SAS :

`CALL SYMPUT` (*nomVar*, *valeurNomVar*) ; Dans une étape data

Créer une ou plusieurs macro-variables en leur affectant les valeurs d'une variable d'une table SAS.

Accès globaux :

`%put _ALL_` : toutes les macro-variables

`%put _AUTOMATIC_` : macro-variables automatiques

`%put _USER_` : macro-variables créées par l'utilisateur

Les macros-fonctions

- ❑ `%index(&mvar,ab)` : recherche la chaîne de caractères “ab” dans la macro-variable *mvar*
- ❑ `%length(&mvar)` : retourne la longueur de la macro *mvar*
- ❑ `%scan(&mvar,n,delim)` : retourne le nième mot de *mvar* (3ème argument optionnel : délimiteur)
- ❑ `%substr(&mvar,i,n)` : extrait n caractères à partir du ième dans le contenu de *mvar*
- ❑ `%upcase(&mvar)` : retourne le contenu de *mvar* en majuscule
- ❑ `%eval(expression)` : évalue le calcul sur des entiers à partir de macro-variables
- ❑ `%sysevalf(expression, type_conversion)` : évalue des calculs sur des valeurs numériques décimales à partir d'expression contenant une macro-variable
- ❑ `%sysfunc(Fonction(argument(s),<format.>)` : utilisation des fonctions SAS

```
60 %put %scan(&semaine,3); *Affiche mercredi;
```

```
51 %let i=3+4;
52 %put &i; *i contient 3+7;
53 %let i=%eval(3+4);
54 %put &i; * i contient 7;
```

```
66 %let nb = 25 ;
67 %let racine2 = %sysfunc(sqrt(&nb),8.2) ;
68 %put &racine2; *Affiche 5;
```

Les macros-programmes

Macros programmes :

- Déclaration :
 - `%macro nomMacro(paramètres);`
 ... (*code SAS*)
 - `%mend;`
- Appel: `%nomMacro(paramètres);`

Boucles et conditions :

- `%if expression %then texte ;%else texte ;`
- `%do indice = début %to fin %by increment ; instructions %end ;`
- `%do %while(expression) ; instructions %end ;`
- `%do %until(expression) ; instructions %end ;`

Exemple de macro programme

```

1 *Création de la macro;
2 %macro recup_espece(nom_espece="", long_petal=0);
3   proc sql noprint;
4     select count(*) into :nbligne
5     from iris
6     where Species like "&nom_espece";
7   quit;
8   %if &nbligne.>0 %then %do;
9     proc sql noprint;
10      create table caracteristiques_&nom_espece as
11      select *
12      from iris
13      where Species like "&nom_espece" and Petal_Length>&long_petal;
14    quit;
15  %end;
16  %else %do;
17    proc sql;
18      create table caracteristiques_all as
19      select *
20      from iris
21      where Petal_Length>&long_petal;
22    quit;
23  %end;
24 %mend;
25
26 *Déclaration des macro-variables;
27 %let espece=setosa;
28 %let long_petal=%sysevalf(1.2);
29
30 *Appel de la macro;
31 %recup_espece(nom_espece=&espece., long_petal=&long_petal.);
32

```

Si valeur par défaut du paramètre, obligation d'exprimer le nom des paramètres lors de l'appel

Si la macro-variable est de type autre que caractère, ajouter sysevalf()

Ressources



- <http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#macro-stmt.htm> (Documentation SAS officielle)
- <https://dms.umontreal.ca/~langlois/STT1682/Cours%207%20-%20Language%20Macro.pdf>
- http://math.agrocampus-ouest.fr/infoglueDeliverLive/digitalAssets/19613_SAS_macros.pdf

Exportation des données

Syntaxe

Obs.	sepal_length	sepal_width	petal_length	petal_width	variety	binomial_name
1	5.1	3.5	1.4	0.2	Setosa	Iris setosa
2	4.9	3	1.4	0.2	Setosa	Iris setosa
3	6.2	2.9	4.3	1.3	Versicolor	Iris versicolor
4	5.1	2.5	3	1.1	Versicolor	Iris versicolor
5	5.7	2.8	4.1	1.3	Versicolor	Iris versicolor
6	6.2	3.4	5.4	2.3	Virginica	Iris virginica
7	5.9	3	5.1	1.8	Virginica	Iris virginica

	A	B	C	D	E	F
1	sepal_length	sepal_width	petal_length	petal_width	variety	binomial_name
2	5.1	3.5	1.4	0.2	Setosa	Iris setosa
3	4.9	3	1.4	0.2	Setosa	Iris setosa
4	6.2	2.9	4.3	1.3	Versicolor	Iris versicolor
5	5.1	2.5	3	1.1	Versicolor	Iris versicolor
6	5.7	2.8	4.1	1.3	Versicolor	Iris versicolor
7	6.2	3.4	5.4	2.3	Virginica	Iris virginica
8	5.9	3	5.1	1.8	Virginica	Iris virginica

dataset iris_final

iris_final.csv

Procédure **EXPORT**

```

53 PROC EXPORT DATA=iris_final
54     OUTFILE = "/folders/myfolders/iris_final.csv";
55     DELIMITER = ";";
56 RUN;
```

Ressources



- <https://od-datamining.com/knwbase/export-sas-excel-explique-a-fille/> (L'export en classeur Excel)
- <http://support.sas.com/documentation/cdl/en/acpcref/63184/HTML/default/viewer.htm#a003102702.htm> (Documentation SAS officielle PROC EXPORT Statement)