Opérations sur les bases

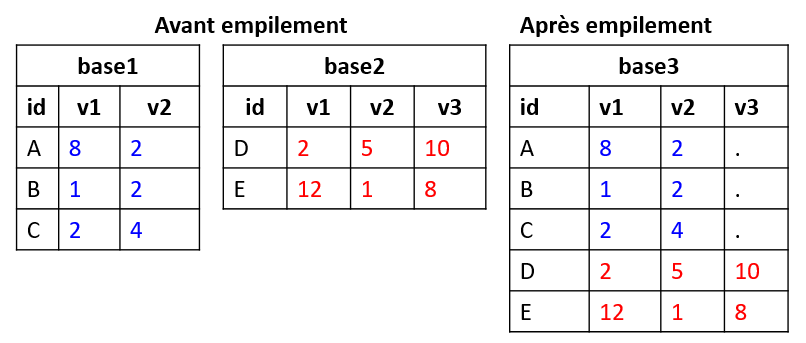
Table of contents

# 1. **Fusion**

* Deux types de fusions:
  + La fusion verticale non controlée - empilement - (append)
  + la fusion horizontale contrôlée - appariement - (merge).

## 1.1 Append

* Consiste simplement à ajouter des observations entre plusieurs bases, avec ou non un même jeu de variables.



On va générer les deux bases de données avec la commande input (non traité dans cette formation: help input).

clear   
input str6 id v1 v2  
 "A" 8 2   
 "B" 1 2   
 "C" 2 4  
end  
  
list  
  
save base1, replace

+--------------+  
 | id v1 v2 |  
 |--------------|  
 1. | A 8 2 |  
 2. | B 1 2 |  
 3. | C 2 4 |  
 +--------------+  
file base1.dta saved

clear   
input str20 id v1 v2 v3  
 "D" 2 5 10  
 "E" 12 1 8   
end  
list  
  
save base2, replace

+-------------------+  
 | id v1 v2 v3 |  
 |-------------------|  
 1. | D 2 5 10 |  
 2. | E 12 1 8 |  
 +-------------------+  
file base2.dta saved

La syntaxe de la commande append consiste à ajouter une ou plusieurs bases à la base active avec l’argument using.

append using base1  
sort id  
list

+-------------------+  
 | id v1 v2 v3 |  
 |-------------------|  
 1. | A 8 2 . |  
 2. | B 1 2 . |  
 3. | C 2 4 . |  
 4. | D 2 5 10 |  
 5. | E 12 1 8 |  
 +-------------------+

On peut sélectionner les variables de la base qui sera empilée à la base active avec l’option keep.  
Dans l’exemple, si la base active est *base1*, on peut ne pas vouloir ajouter la variable *v3* seulement renseignée pour les observations de *base2*.

use base1, clear  
append using base2, keep(id v1 v2)  
list

(variable id was str6, now str20 to accommodate using data's values)  
  
 +--------------+  
 | id v1 v2 |  
 |--------------|  
 1. | A 8 2 |  
 2. | B 1 2 |  
 3. | C 2 4 |  
 4. | D 2 5 |  
 5. | E 12 1 |  
 +--------------+

Si les informations précédentes étaient ventilées dans trois bases, une par variable *v*, et avec le même niveau d’observation (A,B,C,D,E dans les 3 bases), l’utilisation de append conduirait à une structure empilée non souhaitable avec une réplication des *id*.

Pour obtenir la base finale proprement appariée, il convient de faire une fusion horizontale contrôlée par une une clé d’identification.

## 1.2 Merge

Stata demande que les bases soient soit triées (sort) sur la clé d’appariement en amont de l’opération. Sinon un message d’erreur sera renvoyé.  
:::

* La base active (ouverte) est appelée **base master**
* La base qui sera appariée à la base ouverte est appelée base **using** [[1]](#footnote-24)

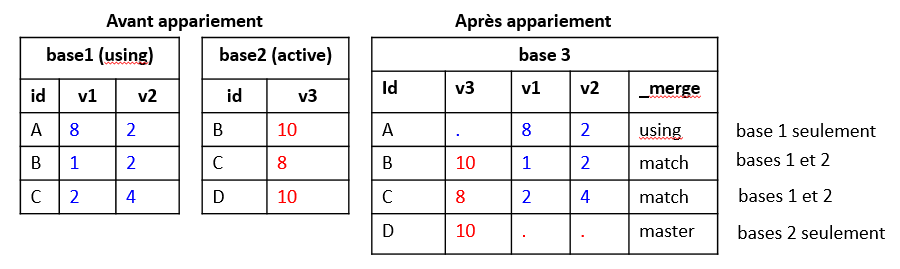
**Syntaxe minimale 1 avec préfixe**:

merge [1:1] [1:m] [m:1] id\_var using nom\_base

* Ici on peut apparier plus de deux bases.
* On ne dispose pas de sécurité, si les niveau d’identification sont différents.

### 1.2.1 Même niveau d’identification

Partons des informations suivantes: - *Base1* comprend la variable d’identification *id* (observations A,B,C) et de deux variables numériques *v1* et *v3* - *Base2* comprend la même variable d’identification *id* (observations B,C,D) et de la variable numérique *v3*

 Le niveau d’identification est identique dans les deux bases. Il s’agit donc d’un **merge 1:1** [one to one]

On va de nouveau générer les bases avec input.

clear   
input str1 id v1 v2   
"A" 8 2   
"B" 1 2  
"C" 2 4   
end  
list  
  
sort id  
save base1, replace

+--------------+  
 | id v1 v2 |  
 |--------------|  
 1. | A 8 2 |  
 2. | B 1 2 |  
 3. | C 2 4 |  
 +--------------+  
file base1.dta saved

Rappel: bien faire le sort sur la base using

clear  
input str1 id v3   
"B" 10   
"C" 8  
"D" 10   
end  
list  
  
sort id  
save base2, replace

+---------+  
 | id v3 |  
 |---------|  
 1. | B 10 |  
 2. | C 8 |  
 3. | D 10 |  
 +---------+  
file base2.dta saved

merge 1:1 id using base1

Result Number of obs  
 -----------------------------------------  
 Not matched 2  
 from master 1 (\_merge==1)  
 from using 1 (\_merge==2)  
  
 Matched 2 (\_merge==3)  
 -----------------------------------------

* L’output affiche le résultat de l’appariement à l’aide d’un tableau.
* Une variable, **\_merge**, est générée. Elle permettra de filtrer si nécessaire les observations selon le résultat de l’aapriement. Contrairement à d’autres application, cette opération n’est pas effectuée en amont avec des fonctions où des options spécifiques. Par exemple avec R: left\_join, right\_join, inner\_join. \_merge = 1 : observations qui se trouvent seulement dans la base active (master) \_merge = 2 : observations qui se trouvent seulement dans la base using (appariée) \_merge = 3 : observations communes aux bases master et using.
* Les variables de la base master/active sont positionnées en tête.

sort id  
list

+-------------------------------------+  
 | id v3 v1 v2 \_merge |  
 |-------------------------------------|  
 1. | A . 8 2 Using only (2) |  
 2. | B 10 1 2 Matched (3) |  
 3. | C 8 2 4 Matched (3) |  
 4. | D 10 . . Master only (1) |  
 +-------------------------------------+

Si on souhaite seulement conserver les observations communes aux deux bases (\_merge=3):

keep if \_merge==3  
list

(2 observations deleted)  
  
 +---------------------------------+  
 | id v3 v1 v2 \_merge |  
 |---------------------------------|  
 1. | B 10 1 2 Matched (3) |  
 2. | C 8 2 4 Matched (3) |  
 +---------------------------------+

|  |
| --- |
| Variable \_merge et appariement successif |
| Pensez à supprimer la variable \_merge si plusieurs opérérations d’appariement sont effectués. La commande ne prévoit pas d’écraser la variable du merge précédent. |

**Situation avec plus d’une base à apparier**

On ne peux pas utiliser la syntaxe avec préfixe (ici merge 1:1).

On va ajouter une nouvelle base qui sera appariée avec les deux premières, qui seront donc les deux bases de type *using*.

clear  
input str1 id str3 v4   
"A" "Non"   
"B" "Oui"   
"C" "Oui"   
end  
  
list   
  
sort id

+----------+  
 | id v4 |  
 |----------|  
 1. | A Non |  
 2. | B Oui |  
 3. | C Oui |  
 +----------+

merge id using base1 base2  
  
order id v1 v2 v3 v4 \_merge1 \_merge2 \_merge   
  
list

(you are using old merge syntax; see [D] merge for new syntax)  
  
 +------------------------------------------------------+  
 | id v1 v2 v3 v4 \_merge1 \_merge2 \_merge |  
 |------------------------------------------------------|  
 1. | A 8 2 . Non 1 0 3 |  
 2. | B 1 2 10 Oui 1 1 3 |  
 3. | C 2 4 8 Oui 1 1 3 |  
 4. | D . . 10 0 1 2 |  
 +------------------------------------------------------+

On obtient maintenant 3 variables \_merge:

* \_merge1. Donne le résultat de l’appariement entre la nouvelle base et *base1*: 0 si seulement dans une seule des deux bases (D), 1 si dans les deux bases (A,B,C).
* \_merge2. Donne le résultat de l’appariement entre la nouvelle base et *base2*: 0 si seulement dans une seule des deux bases (A,D), 1 si dans les deux bases (B,C).
* \_merge. Résume rapidement le matching entre les bases: on retrouve au moins une fois les observations (A,B,C) dans l’un des deux appariement (\_merge=3), on trouve une observation (D) qui ne se trouve que dans une base *using* (\_D\_merge=2).

Si l’on souhaite conserver les observations communes aux trois bases, on peut sommer les valeurs de \_merge1 et \_merge2 et conserver les observations dont la valeurs de cette somme est égale au nombre d’appariements; ou faire une sélection des observations avec un filtre conditionnel, ici:

keep if \_merge1==1 & \_merge2==1  
list  
  
drop \_merge\*

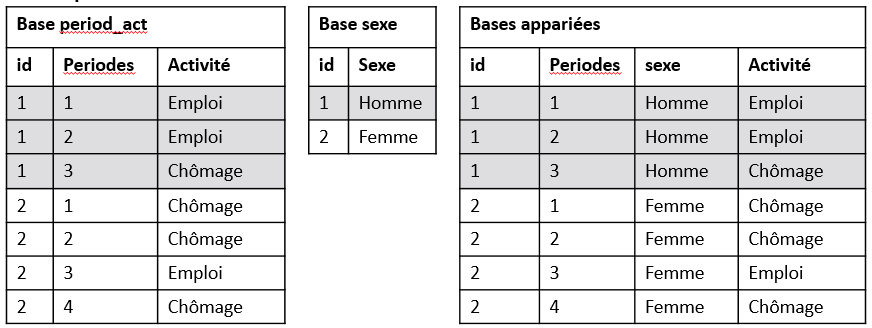
(2 observations deleted)  
  
 +------------------------------------------------------+  
 | id v1 v2 v3 v4 \_merge1 \_merge2 \_merge |  
 |------------------------------------------------------|  
 1. | B 1 2 10 Oui 1 1 3 |  
 2. | C 2 4 8 Oui 1 1 3 |  
 +------------------------------------------------------+

|  |
| --- |
| Commande join du package ftools |
| [A tester]   * [Documentation](https://github.com/sergiocorreia/ftools). * Permet de gagner 70% de durée d’exécution lorsque la volumétrie dépasse 100000 observations * Gère en amont le tri des bases appariée. |

### 1.2.2 Niveaux d’identification différents

Un merge de type 1:1 n’est pas possible. Dans l’exemple qui suit la base *period\_act* liste pour deux personnes le statut d’activité observé pour plusieurs périodes soit des observations multiples pour chaque individus, et la base *sexe* donne une caractéristique unique pour chaque individu. Selon le statut des bases appariée (maste ou using), l’appariement est de type 1:m ou m:1.

* Si la base active est à observations multiple sur la clé d’identification: m:1
* Si la base active est à observation unique sur la clé d’identification: 1:m



On va de nouveau générer les données avec input

clear   
input id périodes str8 Activité  
1 1 "Emploi"  
1 2 "Emploi"  
1 3 "Chômage"  
2 1 "Chômage"   
2 2 "Chômage"  
2 3 "Emploi"  
2 4 "Chômage"   
end   
list   
sort id   
save "period\_act", replace

+--------------------------+  
 | id périodes Activité |  
 |--------------------------|  
 1. | 1 1 Emploi |  
 2. | 1 2 Emploi |  
 3. | 1 3 Chômage |  
 4. | 2 1 Chômage |  
 5. | 2 2 Chômage |  
 |--------------------------|  
 6. | 2 3 Emploi |  
 7. | 2 4 Chômage |  
 +--------------------------+  
file period\_act.dta saved

clear  
input id str6 sexe   
1 "Homme"   
2 "Femme"   
end   
list  
sort id   
save "sexe", replace

+------------+  
 | id sexe |  
 |------------|  
 1. | 1 Homme |  
 2. | 2 Femme |  
 +------------+  
file sexe.dta saved

Si on faisait un merge 1:1, Stata renverrai le message d’erreur suivant:

merge 1:1 id using activités  
  
variable id does not uniquely identify observations in the using data  
r(459);

Ici la base active est la base *sex*. Le prefixe qui doit être utilisé est donc **1:m** [[2]](#footnote-39)

merge 1:m id using period\_act  
sort id période  
list

Result Number of obs  
 -----------------------------------------  
 Not matched 0  
 Matched 7 (\_merge==3)  
 -----------------------------------------  
  
 +------------------------------------------------+  
 | id sexe périodes Activité \_merge |  
 |------------------------------------------------|  
 1. | 1 Homme 1 Emploi Matched (3) |  
 2. | 1 Homme 2 Emploi Matched (3) |  
 3. | 1 Homme 3 Chômage Matched (3) |  
 4. | 2 Femme 1 Chômage Matched (3) |  
 5. | 2 Femme 2 Chômage Matched (3) |  
 |------------------------------------------------|  
 6. | 2 Femme 3 Emploi Matched (3) |  
 7. | 2 Femme 4 Chômage Matched (3) |  
 +------------------------------------------------+

|  |
| --- |
| Tip |
| Le tri de la base est régulièrement modifié après ce type d’appariement. Penser donc à retrier les données proprement, surtout quand il s’agit comme ici d’informations biographiques (sort id périodes) |

De nouveau les préfixes sont optionnels, et permettent seulement de contrôler l’appariement. On peut sans soucis fusionner des informations contextuelles avec des informations multiples avec seulement merge.

use sexe, clear  
merge id using period\_act  
sort id périodes  
list

(you are using old merge syntax; see [D] merge for new syntax)  
variable id does not uniquely identify observations in period\_act.dta  
  
 +-------------------------------------------+  
 | id sexe périodes Activité \_merge |  
 |-------------------------------------------|  
 1. | 1 Homme 1 Emploi 3 |  
 2. | 1 Homme 2 Emploi 3 |  
 3. | 1 Homme 3 Chômage 3 |  
 4. | 2 Femme 1 Chômage 3 |  
 5. | 2 Femme 2 Chômage 3 |  
 |-------------------------------------------|  
 6. | 2 Femme 3 Emploi 3 |  
 7. | 2 Femme 4 Chômage 3 |  
 +-------------------------------------------+

1. Cela peut être plusieurs bases. [↑](#footnote-ref-24)
2. **m:1** renvoit un message d’erreur. Dans ce sens, la base active doit être *period\_act* et la base using *sexe*. [↑](#footnote-ref-39)