Notas de clase – Taller de Stata[1]

Clase 5 - Unión y compresión de bases de datos

Contenido

1. append: adición de nuevas observaciones
2. merge: adición de nuevas variables
3. collapse: agregación de bases de datos
4. contract: bases de datos con frecuencias
5. reshape: transformación entre formas wide y long

[Estas notas se diseñaron con base en la versión 16 de *Stata*, conforme el software cambie las notas serán actualizadas].

[1]Estas notas están basadas en la guía desarrollada previamente para este curso por Rodrigo Azuero Melo, Nicolás de Roux, Luis Roberto Martínez, Román Andrés Zárate y Santiago Gómez Echeverry.

Taller de Stata
Clase 5 – Unión y compresión de bases de datos
Miguel Garzón Ramírez, Cristhian Acosta Pardo
Facultad de Economía, Universidad de los Andes

En este texto exponemos los aspectos básicos para el procesamiento de bases de datos. Empezamos con la adición de nuevas observaciones por medio del comando *append*, se explican los diferentes tipos de adición de nuevas variables por medio del comando *merge*, la creación de bases de datos con información agregada por medio del comando *collapse,* la generación de bases de datos con información de estadísticas descriptivas por medio de comando *contract* y, finalmente, la transformación de bases de datos basadas en individuos a bases de datos basadas en observaciones en el tiempo de los individuos por medio del comando *reshape*.

1. append: adición de nuevas observaciones

La unión de dos bases de datos que contienen las mismas variables consiste en la unión de sus observaciones. El comando *append* agrega las nuevas observaciones después de la última observación de la base de datos actual. La base de datos que está cargada en la memoria de *Stata* es llamada base de datos *master*, la base de datos con las observaciones a agregar es llamada *using*.

La sintaxis del comando es la siguiente:

Ej. 1 ⟶ append using *filename* [, options]

Si las bases de datos no tienen las mismas variables, estas se agregan asignando *missing values* para las observaciones con datos faltantes según sea el caso. La opción más común en el uso del comando *append* es *generate(newvar)*, esta permite crear una variable que muestra si la observación viene de la base *master* o de la base *using*. Las demás opciones permiten no copiar las etiquetas o las notas o evitar el error que surge de tratar de unir bases de datos donde una variable contiene un dato numérico y en la otra el dato es una cadena de caracteres.

En clase disponemos de una base de datos que contiene una relación de los códigos y nombres de departamentos con los códigos y nombres de municipios. Se quiere agregar algunos municipios que hacen falta en la base de datos. Utilizamos el comando *append* de la siguiente manera:

Ej. 2 ⟶
```
use "Base_Depto_Mun", replace   //Cargar la base master
append using "Base_Depto_Mun_2", gen(a1) //Anexar la base using
                                 creando una variable que marca la
                                 fuente de las observaciones
```

2. merge: adición de nuevas variables

El comando *merge* permite agregar variables a la base de datos *master*. Las bases *master* y *using* tienen información sobre las observaciones y se va a reunir en una sola base de datos. A continuación, veremos la sintaxis básica del comando *merge*, algunas de sus opciones y los tipos de uniones.

Cada observación debe identificarse con un código único en su variable llave (*key* o *matching variable)* a través de la cual se hace la unión. Preferiblemente se debe utilizar como variable llave aquellas que contengan nombres o datos con formato estándar, como cadenas de caracteres. Algunos ejemplos de variables que pueden ser usadas como llaves son el número de la cédula, el número de identificación institucional, códigos de usuario, códigos de cuentas, entre otros. También se puede usar una combinación de variables para hacer la unión, como código de país y año.

Antes de hacer la unión se debe asegurar que las bases de datos no tengan observaciones repetidas o duplicados. Si la o las variables llave tienen duplicados es muy probable que la operación *merge* genere errores, ya que se puede emparejar información a observaciones que no le corresponden o simplemente la operación no se ejecuta. Esta verificación se hace con el comando *duplicates*[2]. La sintaxis de este comando es:

Ej. 3 ⟶ duplicates list *varlist* // *varlist* es el conjunto de variables sobre las cuales se quieren identificar los duplicados, normalmente son la o las variables que van a ser llave.

La sintaxis básica del comando *merge* para esta operación es la siguiente:

Ej. 4 ⟶ merge 1:1 varlist using filename [, options]

En clase, a la base de datos con códigos y nombres de departamentos y municipios vamos a agregar información de población a nivel municipal. En este caso, el comando sería:

Ej. 5 ⟶ merge 1:1 cod_mpio using "Poblacion_total", gen(m1)

[2]Puede revisar la documentación de este comando en: https://www.stata.com/manuals/dduplicates.pdf

En donde *1:1* indica el tipo de unión a realizar. *cod_mpio* es la variable llave. Esta variable debe tener el mismo nombre en ambas bases de datos, *master* y *using*. *"Poblacion_total"* es el nombre de la base de datos *(using)* con la nueva información.

Al hacer la unión, *Stata* crea por defecto una nueva variable (llamada *_merge*) que identifica con el número *3* si la observación fue encontrada en las dos bases de datos y fue unida (se hizo *match*), con el número 1 si alguna observación en *master* no encontró *match* en *using*, y con el número 2 si alguna observación en *using* no encontró *match* en *master*. La opción *gen()* permite cambiar el nombre de la nueva variable que informa el resultado de la operación por observación, esto es conveniente cuando se hacen varios *merge* de manera sucesiva.

Existen otras opciones del comando *merge*. En principio, datos de la base *master* no se modifican ya que la base using contiene nuevas variables. Sin embargo, es posible que la base *using* contenga variables en común y se desee que esta información haga parte de la base de datos final por estar mas actualizada. Hay dos formas de modificar la informacion de *master* con *using*. La primera es actualizar la información de *master*, esto es que si las variables en común tienen *missing values* en *master*, estos serán reemplazados con la información disponible en *using* con la opción *[, update]*. La segunda es reemplazar la información de toda la variable en *master* por la misma variable en *using* con la opción *[, replace]*. Dado esto, la variable creada por defecto *_merge* tomará el valor *4* si en esa observación se hizo una actualización de la información y el valor *5* si en esa observación se reemplazó la información.

La unión del ejemplo 5 es del tipo 1:1 porque las observaciones de ambas bases de datos están a nivel municipal, la misma unidad de observación. Entonces, una observación en *master* corresponde directamente a una observación en *using*. Hay casos donde, por ejemplo, se tiene información a nivel municipal y debe agregarse información a nivel departamental. Esto implica otro tipo de unión llamado *muchos a uno (m:1)*, teniendo en cuenta que varios municipios corresponden a un solo departamento. En la práctica, el dato de un departamento va a ser colocado en todos los municipios que correspondan a ese departamento. En nuestro caso de clase, vamos a agregar información de la cantidad de parques naturales por departamento a cada observación municipal.

Ej. 6 ⟶ merge m:1 cod_dpto using "natdis.dta", gen(m2)

En esta sintaxis, *m:1* denota el tipo de unión, el término previo a (:) indica que a varios valores de la base de datos *master* les corresponde el mismo valor de la base *using*, la cual se indica con el término posterior a (:). La variable llave es *cod_dpto*, ya que corresponde al nivel de desagregación de la base *using*. Es importante tener en cuenta que esta variable debe contener valores únicos (no duplicados) en *using* y valores duplicados en *master,* ya que puede haber varios municipios en un departamento; esta es otra forma de entender la notación *m:1*. En la práctica, para nuestro ejemplo de clase, todos los municipios cuyo código de departamento sea 5 (Antioquia) van a tener en la nueva variable *natdis* el valor 43. El último tipo de unión, *uno a muchos (1:m)*, es homólogo al anterior.

Antes de hacer una unión es importante realizar un análisis de la lógica utilizada para unir los datos. *Stata* cuenta con un tipo de unión *muchos a muchos (m:m)*. Esta forma de unión es inconveniente por cuanto no se ajusta a una lógica clara de unión de datos, no es fácil determinar cuáles datos de *using* se están colocando en cuáles observaciones de *master*, por lo tanto, no se recomienda su uso. En la práctica, bajo un razonamiento lógico estructurado la unión *m:m* no es necesaria.

3. collapse: agregación de bases de datos

Este comando sirve para crear bases de datos con información agregada. En nuestro caso, sirve para crear un base de datos con información agregada a nivel departamental con base en la información disponible a nivel municipal. Por ejemplo, queremos construir una base de datos con contenga la información de la población total por departamento. Para esto debemos sumar la población de los municipios de cada departamento. La sintaxis básica de este comando es:

Ej. 7 ⟶ collapse [(stat)] varlist1 [, by(varlist2)]

Donde *[(stat)]* indica el tratamiento numérico de una o más variables denotadas por *varlist1*. Esto es, por ejemplo, si estamos sumando, calculando la media, algún percentil o algún otro tratamiento a la información de población municipal para agregarla a nivel departamental. En otras palabras, *stat* hace referencia a una estadística descriptiva por medio de la cual se quiere

agregar la base de datos. En la documentación del comando *collapse*[3] se pueden consultar todos los estadísticos disponibles. Es posible calcular varias estadísticas de la misma variable al interior del mismo *collapse* colocando un nombre distinto a cada una en la nueva base de datos agregada.

Finalmente, *[, by(varlist2)]* índica el nivel de agregación de la operación. Si *varlist2* contiene una sola variable, la unidad de agregación de las observaciones de la nueva base de datos corresponderá a los valores únicos de esta variable. Por ejemplo, si la variable de agregación es el código del departamento, las observaciones de la nueva base van a ser cada uno de los departamentos existentes en la base de datos. Si *varlist2* contiene varias variables, las observaciones de la nueva base de datos serán todas las posibles combinaciones de los valores únicos de estas variables. Por ejemplo, si las variables de agregación son el código del departamento y el año las nuevas observaciones serán cada departamento en cada uno de los años.

Si no se incluye ningún *[, by],* la base de datos tendrá una única observación de estadísticas descriptivas agregadas. Tenga en cuenta que *varlist2* (en *[, by]*) puede incluir variables *string,* pero *varlist1* no, ya que *Stata* no puede calcular estadísticas de variables *strings* pero sí definir categorías para las mismas. Es importante que *varlist2* contenga variables como códigos o años, cuyo formato sea homogéneo. Utilizar nombres, cadenas de caracteres de longitud dispersa o caracteres especiales puede llevar a resultados no deseados en la operación o que simplemente no se ejecute.

Para nuestro ejemplo de clase, queremos construir una base de datos que muestre la población agregada a nivel departamental. En la práctica la sintaxis sería la siguiente:

Ej. 8 ⟶ collapse (sum) pob_total, by(cod_dpto)

Este comando crea una nueva base de datos cuyas variables son *cod_dpto* y *pob_total*. La variable *pop_total* contiene la población departamental, calculada como la suma de la población municipal. Suponga que se quiere construir una base de datos con la población municipal promedio por departamento. La sintaxis de este caso puede ser:

[3] Disponible en https://www.stata.com/manuals/dcollapse.pdf

Ej. 9 ⟶ collapse (mean) pob_total, by(cod_dpto)

collapse pob_total, by(cod_dpto)

Las dos opciones crean la misma base de datos, el comando *collapse* toma por defecto el tratamiento estadístico de calcular la media y por lo tanto no es necesario incluirlo en la sintaxis, aunque preferible.

Suponga que tenemos datos de la población y de la cantidad de parques naturales a nivel municipal y por año. Si necesitamos una base de datos que contenga, por departamento y año, el promedio de población municipal, el total de la población, y el número máximo de parques naturales, la sintaxis sería la siguiente:

```
Ej. 10 ⟶ collapse    avg_pop_mun=pop_total ///
                     (sum) pop_total ///
                     (max) natdis, by(cod_dpto anno)
```


En esta línea de código utilizamos dos veces la variable *pop_total* de la base de datos inicial para crear dos variables de la nueva base de datos. Para hacer esto se debe asignar otro nombre a por lo menos una de las nuevas variables creadas, como se hace con la primera variable *avg_pop_mun*. Después de *[(stat)]* se puede colocar más de una variable y sobre estas se aplicará el mismo tratamiento. Con *by(cod_dpto anno)* las observaciones de la nueva base de datos quedan de la siguiente manera:

departamento 1, año 1
departamento 1, año 2
...
departamento 2, año 1
…

4. contract: bases de datos con frecuencias

El comando *contract* permite colapsar una base de datos en el caso de que se deseen obtener las frecuencias o porcentajes de los valores de una variable o de una combinación de variables. En esencia, este comando crea una nueva base de datos con la misma información que provee el

comando *tab,* con la ventaja de que es fácilmente exportable a Excel o en texto plano. Por defecto, el comando arroja una base de datos de frecuencias, aunque existen opciones para especificar porcentajes o frecuencias y porcentajes acumulados. La sintaxis general es:

contract varlist [if] [in] [weight] [, options]

5. reshape: transformación entre formas wide y long

En una base de datos es importante definir qué se entiende por observación. En algunos casos, una observación puede ser un individuo y la información que se posee de él son datos de sus ingresos en diferentes momentos del tiempo, a estas bases *Stata* las llama *wide*. En otros casos, una observación puede ser un individuo en un momento determinado, estas bases tienen la forma de datos panel y se llaman *long*. El comando *reshape* puede convertir una base de datos de *wide* a *long* y viceversa.

La esencia de esta operación es que, al crear la base de datos *long* se crea una variable de tiempo, que normalmente contiene datos de años. En nuestro ejemplo, se puede entender como el ingreso de una persona en el año *2001*, *2002*, *2003*... La base de datos *wide* puede contener variables llamadas *ingreso2001*, *ingreso2002*, *ingreso2003*, la parte numérica de estos nombres va a ser la información contenida en la nueva variable de tiempo *(j)* en la nueva base de datos *long*. De esta forma, se entiende que es necesario estandarizar los nombres de las variables en la base de datos *wide* antes de realizar la conversión a *long*.

La documentación de *Stata* tiene ejemplos ilustrativos para entender cómo funciona el comando y nos vamos a permitir utilizarlos en este espacio. La sintaxis general de este comando es la siguiente, para convertir de *wide* a *long* y de *long* a *wide* respectivamente.

- reshape long stubnames , i(varlist) j(varlist) options
- reshape wide stubnames , i(varlist) j(varlist) options

En donde *long/wide* denotan la forma de la nueva base de datos, *stubnames* identifica a las variables que contienen los identificadores de tiempo, *i(varlist)* denota el nombre de la o las variables que identifican de manera única a las observaciones en una base de datos *wide*, *j(varlist)* identifica a la

variable que contiene o contendrá la información de tiempo. *j(varlist)* es creada cuando se crea una base de datos *long* y es eliminada cuando se crea una base de datos *wide*.

En la práctica, la pequeña base que se muestra a continuación puede ser convertida a *long* por medio del siguiente comando

Ej. 11 ⟶ reshape long stub, i(i) j(j)

wide

i	stub1	stub2
1	4.1	4.5
2	3.3	3.0

long

i	j	stub
1	1	4.1
1	2	4.5
2	1	3.3
2	2	3.0

Donde *long* indica que estamos creando una base de ese tipo, *stub* es la parte común del nombre de las variables que contienen el indicador de tiempo. *i(i)* identifica de manera única a las observaciones en *wide* y *j(j)* crea la variable que indica el tiempo y toma sus valores de la parte variable de los nombres de las variables *stub*.

(wide form)

i		..... x_ij		
id	sex	inc80	inc81	inc82
1	0	5000	5500	6000
2	1	2000	2200	3300
3	0	3000	2000	1000

(long form)

i	j		x_ij
id	year	sex	inc
1	80	0	5000
1	81	0	5500
1	82	0	6000
2	80	1	2000
2	81	1	2200
2	82	1	3300
3	80	0	3000
3	81	0	2000
3	82	0	1000

En este nuevo caso tenemos un conjunto de variables que tiene al final de sus nombres un componente que indica tiempo. Adicionalmente, tenemos una variable (*sex*) que no varía en el tiempo. Los datos de la variable *sex* se repiten para cada combinación de los valores únicos de individuo y cada año, esto es, incluirlos como información del individuo en las observaciones

que correspondan de la forma *long*. Para convertir de la forma *wide* a la forma *long* utilizamos la siguiente línea de código:

Ej. 12 ⟶ reshape long inc, i(id) j(year) string

Note que se ha agregado una nueva opción después de *j(varlist),* en este caso *string* le indica a *Stata* que la nueva variable de tiempo se cree con formato de cadena de caracteres. Si después de haber hecho la conversión de *wide* a *long* quiere transformar su base de *long* a *wide* hay dos formas, la implícita y al explícita, de la siguiente manera:

Ej. 13 ⟶ reshape wide //comando implícito

Ej. 14 ⟶ reshape wide inc, i(id) j(year) string //comando explícito