

17.871

Primavera de 2002

Introducción al programa Stata¹

1. Introducción

Stata es una aplicación global y actual, basada en comandos, para el análisis estadístico, la gestión de datos y de gráficos. Existen versiones para PC/DOS/Windows, Mac, y Unix. Aquí vamos a repasar brevemente algunos de los elementos más importantes del programa, reproducir una sesión de muestra y, por último, presentamos una tabla con algunos de los comandos principales de Stata. El presente documento se elaboró para introducir Stata en el entorno operativo Athena, por lo tanto, el sistema operativo que asumimos es Unix.

1. Inicio de Stata

En una ventana xterm teclee lo siguiente:

```
add stata
xstata
```

De este modo se iniciará la interfaz gráfica de Stata. Si, en lugar de teclear `xstata`, teclea `stata`, obtendrá simplemente una interfaz de línea de comandos.

ATENCIÓN: el pasado año, cuando se añadió la interfaz gráfica para Unix, se detectaron algunos problemas con la actualización de Athena. En concreto, `xstata` hacía que a veces el proceso en las máquinas de Athena se detuviera sin remedio y era necesario reiniciar. El problema parece tener que ver con los gestores de ventanas no estándar. La manera de evitar este problema se resume en los tres puntos siguientes:

- (1) Guarde el trabajo con frecuencia.
- (2) Si observa que Athena se cuelga cada vez que utiliza Stata, intente utilizar el gestor de ventanas estándar de Athena.
- (3) Si desea erradicar por completo estos problemas, utilice la versión de línea de comandos de Stata.

1.1 Introducción y edición de comandos

¹La mayor parte de este documento proviene de otro similar creado por Jeroen Wessie, Departamento de sociología, Universidad de Utrecht.

Los comandos se introducen y se editan con el teclado. Los comandos anteriores se almacenan en un buffer y pueden restaurarse para editar. Además, la interfaz gráfica de usuario presenta la ventana “review” que registra los últimos comandos introducidos. Una vez introducido un comando, puede ir a la ventana Review y hacer doble clic en cualquier comando para ejecutarlo de nuevo. Con un solo clic, el comando aparecerá en la ventana Command en donde se puede editar y ejecutar.

Esta es una lista de los comandos de edición más útiles en Unix.

Comando	Tecla
recupera el comando anterior	Ctrl-R
comando siguiente	Ctrl-B
cursor hacia atrás	flecha izq.
cursor hacia adelante	flecha dcha.
mover cursor al principio de la línea	Inicio
mover cursor al final de la línea	Fin
borrar carácter a la izquierda	Retroceso
borrar carácter en la posición del cursor	Supr
borrar línea completa	Ctrl-U
ejecutar comando	Intro

Para consultar estos diez comandos, teclee `#review 10`.

1.2 Salida de Stata

Para salir, utilice el comando `exit`. Si ha trabajado con un conjunto de datos, lo más probable es que haya modificado la información, por ejemplo, creando nuevas variables. Si no guardó antes los datos, Stata impedirá que salga del programa, es un sistema de protección que trata de evitar despistes del usuario. Puede salir de Stata sin guardar los cambios tecleando `exit, clear`.

1.3 Ayuda

Stata utiliza la tecla F1 para el comando `help`. Mediante este comando puede obtener información detallada sobre la mayoría de las características y comandos del programa. Por ejemplo, si teclaea `help regress` obtendrá información sobre como ejecutar regresiones lineales con el comando `regress`.

1.4 Búsqueda

El comando `search topic` permite buscar el comando Stata para análisis con referencia a `topic`. Por ejemplo, `search regression` proporciona una visión concisa de los comandos relativos al análisis de regresión. Nota: al final de la sección de ayuda, encontrará también una lista de comandos relacionados.

1.5 Stata en Internet

El sitio de Stata en Internet es www.stata.com, y podemos acceder a él desde Stata con el menú Help. En el sitio web, hallará una pagina FAQ muy completa y diversos artículos.

1.6 Identificadores

Un identificador (“nombre”), como el nombre de un comando o variable, tiene como máximo 32 caracteres (incluidos mayúsculas y minúsculas, dígitos y el guión bajo), y es *preferible* que el primer carácter sea una letra. (Este límite de 32 caracteres es nuevo en la versión 7. Las versiones anteriores imponían un límite de 8 caracteres. Por razones de compatibilidad, seguiremos utilizando el límite de 8 caracteres y les pedimos que hagan lo mismo). Stata es *sensible a caja*. Casi todos los comandos están en minúscula.

1.7 Abreviaturas

Como regla general, en Stata es posible abreviar comandos y variables con tal de que el programa entienda el significado. Por ejemplo, si tiene las variables `income1` y `inkvar2` en el conjunto de datos, Stata reconocerá que `inc` es la variable `income1`, pero el programa no será capaz de distinguir si `in` significa `income1` o `inkvar1`. Si tiene intención de especificar todas las variables que comienzan por “in,” puede utilizar la expresión comodín (`in*`).

1.8 Archivos de registro

El comando `log using filename` especifica que todos los comandos que se introduzcan con el teclado, así como los resultados que producen, se guarden en un archivo llamado `filename.scml`. De este modo,

puede guardar resultados y consultarlos. Por desgracia (en mi criterio), Stata v. 7 produce por defecto un archivo *log* que es una versión mutante de un html. Por lo tanto, para convertir este archivo en algo más fácil de leer, tendrá que introducir el comando `translate`. Puede saltarse este paso tecleando `log using filename, text`.

1.9 Comandos Shell

Puede introducir un comando Unix precediéndolo del carácter “!”. Por ejemplo, `!ls` hará un listado de los archivos que se encuentran en el directorio en uso.

1.10 Archivos Batch

Puede crear un archivo de comandos Stata para ejecutarlo en modo batch, lo que resulta muy útil cuando se desea replicar los análisis. En algunos problemas y en el proyecto final tendrá que crear archivos de este tipo para documentar el trabajo. Para crear y probar uno de estos archivos puede utilizar un editor ASCII (como emacs) o el editor de archivos-do en la interfaz gráfica.

1.11 Sesión de muestra

A continuación presentamos una breve sesión introductoria en Stata con el ejemplo de los cargos públicos negros elegidos (`beo_example.dta`) data set in `/mit/17.801/Examples`. Los comentarios van precedidos de *.

* Nota: en primer lugar hay que teclear “log using example”.

```
-----
log: /afs/athena.mit.edu/course/17/17.801/Examples/example.smcl
log type: smcl
opened on: 12 Feb 2001, 14:09:37
```

```
. clear
```

* utilice el archivo de sistema existente `/mit/17.801/Examples/black_officials.dta`.

```
. use black_officials
```

* averigüe lo más básico acerca de la organización de este conjunto de datos.

```
. describe
```

```
Contains data from black_officials.dta
```

```

obs:          50
vars:         4                      12 Feb 2001 14:05
size:        900 (99.8% of memory free)

```

```

-----
      storage  display  value
variable name  type    format  label    variable label
-----
state          str2    %9s
beo            float   %9.0g
bpop           float   %9.0g
south         float   %9.0g
-----

```

Sorted by:

* consulte un listado abreviado de los nombres de las variables.

```

. ds
state  beo    bpop    south

```

* consulte las estadísticas resumidas de todas las variables.

```

. summarize

```

Variable	Obs	Mean	Std. Dev.	Min	Max
state	0				
beo	50	.28	5.276246	-9	10.8
bpop	50	6.688	10.53152	-9	30.8
south	50	.22	.418452	0	1

* En el listado anterior, vea como beo y bpop tienen valores mínimos de -9, lo cual
* es imposible (se trata de porcentajes). -9 es un placeholder para estados en los
* que faltan los valores de esas variables. Recodificaremos estas variables para reemplazar
* los valores -9 con el código de valor del sistema que falta.

```

. replace beo=. if beo==-9
(9 real changes made, 9 to missing)

```

```

. replace bpop=. if bpop==-9
(9 real changes made, 9 to missing)

```

* Observemos ahora las estadísticas resumidas teniendo en cuenta los valores que faltan.

```

. summarize

```

Variable	Obs	Mean	Std. Dev.	Min	Max
----------	-----	------	-----------	-----	-----

state		0				
beo		41	2.317073	3.236117	0	10.8
bpop		41	10.13171	8.266633	1.2	30.8
south		50	.22	.418452	0	1

* Una versión más detallada de las estadísticas resumidas, incluidos cuantiles y estadísticas de orden superior.

. summ,detail

state

no observations

beo

Percentiles		Smallest		
1%	0	0		
5%	0	0		
10%	0	0	Obs	41
25%	.2	0	Sum of Wgt.	41
50%	1		Mean	2.317073
		Largest	Std. Dev.	3.236117
75%	2.1	10.4	Variance	10.47245
90%	6.7	10.5	Skewness	1.719571
95%	10.5	10.7	Kurtosis	4.696853
99%	10.8	10.8		

bpop

Percentiles		Smallest		
1%	1.2	1.2		
5%	1.2	1.2		
10%	2.1	1.2	Obs	41
25%	3.3	1.7	Sum of Wgt.	41
50%	7.5		Mean	10.13171
		Largest	Std. Dev.	8.266633
75%	13.6	24.9	Variance	68.33722
90%	22.9	26.6	Skewness	.9372295
95%	26.6	27.7	Kurtosis	2.850443
99%	30.8	30.8		

south

Percentiles		Smallest		
1%	0	0		
5%	0	0		
10%	0	0	Obs	50

25%	0	0	Sum of Wgt.	50
50%	0		Mean	.22
		Largest	Std. Dev.	.418452
75%	0	1		
90%	1	1	Variance	.175102
95%	1	1	Skewness	1.351853
99%	1	1	Kurtosis	2.827506

* Averigüe cuántos estados hay en el sur.

. tabulate south

south	Freq.	Percent	Cum.
0	39	78.00	78.00
1	11	22.00	100.00
Total	50	100.00	

* Haga una lista de los estados del sur (a saber, sur == 1).

. list state if south==1

```

state
1.    AL
4.    AR
9.    FL
10.   GA
18.   LA
24.   MS
33.   NC
40.   SC
42.   TN
43.   TX
46.   VA

```

* Halle el coeficiente de correlación entre beo (porcentaje de la asamblea legislativa del estado que son afroamericanos) y bpop (porcentaje de la población del estado que es afroamericana).

. corr beo bpop south
(obs=41)

	beo	bpop	south
beo	1.0000		
bpop	0.9157	1.0000	

```
south | 0.7597 0.7406 1.0000
```

* Los dos próximos comandos ordenan los datos en función de la variable sur y, a continuación, hallan la correlación entre beo y bpop para los estados del norte (sur == 0) y del sur (sur == 1) por separado.

```
. sort south
```

```
. by south: corr bpop beo
```

```
-> south = 0
(obs=30)
```

```

-----+-----
      |      bpop      beo
-----+-----
 bpop | 1.0000
 beo  | 0.8550 1.0000

```

```
-> south = 1
(obs=11)
```

```

-----+-----
      |      bpop      beo
-----+-----
 bpop | 1.0000
 beo  | 0.9092 1.0000

```

* El comando table informa de las medias de bpop y beo según la región.

```
. table south,c(mean bpop mean beo)
```

```

-----+-----
 south | mean(bpop)  mean(beo)
-----+-----
      0 |          6.47   .8466666
      1 |        20.11818  6.327273

```

* Ejecute la regresión que predice beo en función de bpop y south.

```
. reg beo bpop south
```

Source	SS	df	MS	Number of obs =	41
Model	357.430178	2	178.715089	F(2, 38) =	110.48
Residual	61.4678611	38	1.61757529	Prob > F =	0.0000
				R-squared =	0.8533
				Adj R-squared =	0.8455
Total	418.898039	40	10.472451	Root MSE =	1.2718

beo	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
bpop	.306134	.0362023	8.46	0.000	.2328463	.3794217
south	1.302433	.6671595	1.95	0.058	-.0481606	2.653027
_cons	-1.13402	.3298217	-3.44	0.001	-1.801709	-.4663314

```
* cierre el archivo log.
```

```
. log close
```

```
log: /afs/athena.mit.edu/course/17/17.801/Examples/example.smcl
```

```
log type: smcl
```

```
closed on: 12 Feb 2001, 14:10:35
```

2. Sintaxis de Stata

Stata tiene una sintaxis potente y coherente. Con algunas excepciones, la forma básica es:

```
[by varlist1:] command [varlist2] [weight] [if expr2] [in range] [, options]
```

Ejemplos de comandos Stata:

```
summarize age
regress income educ exp sex
tabulate sex edu if age>25
tabulate sex edu,no freq cell chi2
by cohort: tabulate sex edu
```

Notas sobre la sintaxis:

- El diagrama de sintaxis anterior, las cláusulas opcionales van en corchetes, esto es, [].
- `by varlist1:` solicita un análisis separado de cada modelo de valores de `varlist1:`
- Las cláusulas `[if expr2]` `[in range]` restringen el conjunto de observaciones sobre el cual opera el comando. Pueden darse en cualquier orden.

- Un rango tiene la forma #, o ##, donde # significa un número o l (la letra l), que significa la última observación. Por ejemplo, `in 1/10` significa “sólo para las primeras 10 observaciones”; `in 1` significa “sólo la última observación”; `-5/1` significa “las últimas 5 observaciones”.
 - En este documento no hablaremos sobre pesos.
 - Observe que se requiere una coma antes de las opciones. En la mayoría de los casos hay como máximo una coma. (Algunas pocas funciones separan argumentos con comas).
 - Las líneas que empiezan con * se ignoran. En un archivo .do cualquier texto entre /* y */ se considera un comentario. No tienen que estar en la misma línea: /* */ se puede utilizar para que una nueva línea sea invisible para Stata.
 - Por defecto, las líneas de comando terminan con ENTER (retorno de carro). En un archivo .do (y sólo en este tipo de archivos), se puede cambiar el comando de separación “;” por `#delimiter;` mientras que el comando `#delimiter cr` lo cambia de nuevo a retorno de carro. No se permiten otros caracteres como delimitadores.
 - La sintaxis distingue entre may. y min.: `a` es diferente de `A`. Todos los nombres de Stata van en min.
 - Los nombres pueden tener de 1 a 32 letras, dígitos o guiones bajos, comenzando por una letra. Para facilitar la compatibilidad con versiones anteriores de Stata, es preferible no sobrepasar los 8 caracteres en los nombres.
 - Una variable puede llevar el carácter comodín *. Las listas de variable pueden utilizar - para indicar un rango de variables. En una lista de nombres `v1 - v100` significa `v1 v2, . . . v100`.
 - Stata es compatible con tipos diferentes de variables: enteros (`byte`, `int`, `long`) números reales aproximados (`float`, `double`), fechas y cadenas alfanuméricas. La que está por defecto es `float`.
 - Muchos nombres de sistema comienzan por `_`, por ejemplo:

<code>_n</code>	número de orden de la observación en curso
<code>_N</code>	número total de observaciones
<code>_a11</code>	todas las variables
<code>_b</code>	vector de coeficientes de regresión
<code>_se</code>	vector de errores estándar de coeficientes de regresión
- Combinados con `by`, `_N` y `_n` se refieren al número de observaciones dentro del grupo en curso.
- Para expresiones vea más abajo, o utilice `help exp`. En expresiones lógicas, utilice `==` para igualdad. Stata permite subíndices (con `generate`, sólo en el lado derecho), usando `[1, _n, _N, etc.` Por ejemplo, `gen dx = x - x[_n-1]` significa generar una variable nueva (`dx`) que reste el valor de `x` de la observación anterior del valor de `x` de la observación actual.
 - Las comillas sólo se utilizan en las cadenas. Utilice doble comilla.

3. Expresiones y transformación de datos

3.1 Las expresiones lógicas pueden llevar:

& | > < == ~= >=<=

Aquí (&) es Y, (|) es O, y (~) es NO. Observe el uso del signo igual doble (==) para la igualdad booleana. En Stata un solo = se usa solamente para asignación mientras que el doble == se utiliza para igualdad.

3.2 Las expresiones aritméticas puede llevar:

+ - * / ^ () [] . _n _N

Notas:

- x^y significa x^y . Observe que -2^2 equivale a -4 mientras que $(-2)^2$ equivale a 4 .
- [] sirven para generar subíndices o variables desplazadas: $x[3]$ es el valor de x para la observación 3; $x[_n-1]$ es el valor de x para la observación anterior.
- $_n$ es el número de la observación en curso y N es el número total de observaciones.
- . (el punto) equivale al valor faltante en el sistema cuando se refiere a variables numéricas. (Los valores que faltan de las variables en cadena se representan mediante cadenas vacías). Internamente, los valores que faltan se representan por el mayor valor posible del tipo de datos. Para ello habrá que realizar algunos ajustes. Por ejemplo, $0 <= x$ es cierto si falta x .

Las funciones matemáticas más importantes son:

abs()	valor absoluto.
exp()	la función exponencial, e^0
int()	el entero obtenido mediante truncamiento hacia 0: <code>int(1.1)</code> equivale a 1.
round(x,y)	redondea x en unidades de y . <code>round(x,1)</code> redondea al siguiente entero.
log()	logaritmo natural.
min(varlist)	el mínimo de <i>varlist</i> . Para obtener el mínimo de fila, posiblemente dentro de subgrupos de observaciones, vea la subfunción <code>rmin()</code> de <code>egen</code> .
max(varlist)	el máximo de <i>varlist</i> . Para obtener el máximo de fila, posiblemente dentro de subgrupos de observaciones, vea la subfunción <code>rmax()</code> de <code>egen</code> .
mod(x,y)	x módulo y = el resto cuando x es entero dividido por y .
sqrt()	raíz cuadrada.
sign()	<code>sign(x)</code> equivale a +1, -1, o 0 para $x > 0$, $x < = 0$, o $x == 0$, respectivamente.
sum()	La suma de todos los valores de la expresión () para todas las observaciones previas y la observación en curso (esto es, una suma “simultánea” o “sistemática”).

`uniform()` Esta función genera un número aleatorio uniformemente distribuido entre 0 y 1. No se necesita argumento, pero el () no debe omitirse. La semilla se puede modificar con `set seed`. Por defecto, Stata fija la semilla en el mismo número, generando siempre la misma secuencia de números aleatorios.

Además, `autocode`, `group`, y `recode` son funciones muy útiles para recodificar en un conjunto discreto de valores. El comando `tabulate` se puede utilizar para generar variables ficticias, de las que hablaremos en el siguiente párrafo. También existen varias funciones sobre cadenas y entre cadenas y números, funciones de distribución de la distribución normal, la distribución P^2 (`chiprob(df,x)`), la distribución F , la distribución t y la inversa de la distribución normal. La última función se puede utilizar para generar números aleatorios distribuidos normalmente: `invnorm(uniform())`. Para más información consulte el manual o teclee `help functions`.

3.3 Transformaciones de datos

Stata tiene una extraordinaria capacidad de transformación de datos.

- A diferencia de la mayoría de las aplicaciones estadísticas, existen comandos diferentes para definir nuevas variables (`generate`) y para modificar las ya existentes (`replace`).
- `tabulate` puede utilizarse para generar variables ficticias, que Stata denomina variables de indicador.
- Las expresiones de Stata son bastante potentes: un buen conjunto de funciones y la mezcla de expresiones lógicas y numéricas (ampliaremos esto más adelante).

Para cambiar los valores de una variable discreta también puede utilizar el comando `recode`. Veamos un ejemplo:

```
recode xyz 1=2 2=1 *=3
```

Para la variable `xyz` este comando sustituye 1 por 2, 2 por 1, y todo lo demás por 3. Tres ejemplos de `generate` son:

```
gen laginc = inc[_n-1]
gen loginc = log(inc)
gen hiinc = inc > 100000
```

El primer ejemplo muestra cómo generar una variable desplazada. El tercer ejemplo crea una variable ficticia `hiinc` que equivale a 1 para rentas superiores a 100.000 y para valores faltantes, 0 en caso contrario.

Cualquier expresión lógica se puede utilizar como (parte de) una expresión aritmética. “True” se interpreta como 1; “false” es 0. Y al contrario, toda expresión aritmética se puede interpretar como una expresión lógica. Cualquier expresión que dé como resultado 0 se entiende como “false”; cualquier expresión que dé como resultado otro número o un valor faltante se entiende como “true.”

Asumamos que tenemos una variable `age` (en años) que queremos recodificar en cuatro categorías con puntos de ruptura en 20, 40 y 60 años. La expresión lógica `(age>20)` es 1 (esto es, “true”) para todas las personas mayores de 20. Así, podemos escribir:

```
gen age4 = 1 + (age>20) + (age>40) + (age>60)
```

generando una nueva variable `age4` que es 1, 2, 3 o 4. Es 1 para todos los sujetos de 20 años o menos y 4 para todos los mayores de 60.

Si deseamos transformar una variable continua como `age` en una discreta, utilizando las cotas superiores como nuevos valores podemos utilizar la función recodificar:

```
gen newvar = recode(oldvar, x1, x2, ..., xn)
```

Si `oldvar # x1`, `newvar = x1`, en otro caso, si `x1 < oldvar # x2`, entonces `newvar = x2`, etc.

Para transformar la edad en las mismas cuatro categorías anteriores teclearíamos:

```
gen age4a = recode(age, 20, 40, 60, 80)
```

Ahora `age4a` es justo `20*age4` arriba.

Una versión automática de `recode` es `autocode`.

```
gen newvar = autocode(oldvar, ng, xmin, xmax)
```

Ahora el intervalo `(xmin, xmax)` se divide “automáticamente” en `ng` subintervalos de igual longitud, y el nuevo valor de `newvar` es la cota superior del intervalo al cual `oldvar` pertenece. Observe que la edad superior se ha dividido en intervalos de igual tamaño. Si no tenemos sujetos mayores de 80 años, `age4a` también puede crearse así:

```
gen age4a = autocode(age, 4, 0, 80)
```

Del mismo modo, puede crearse como se indica a continuación una sola variable ficticia que indique todas las observaciones que tengan el valor 3 en la variable:

```
gen x3 = (x==3)
```

donde los paréntesis son opcionales. La expresión lógica `(x == 3)` se utiliza aquí en un contexto numérico, de modo que devuelve el valor 1 si es cierta y 0 si es falsa. Si `x` es discreta y se desea una variable ficticia para cada valor posible, el método anterior es bastante incómodo. Una alternativa mejor es utilizar la opción `generate` de `tabulate` del siguiente modo:

```

                                gen newvar1 = (x==1)
                                gen newvar2 = (x==2)
                                gen newvar3 = (x==3)
tab x, generate(newvar)    es equivalente a los siguientes comandos:

```

... para todos los valores de x

La construcción `by` es de gran utilidad en la manipulación de datos. Por ejemplo, supongamos que tenemos un conjunto de datos con información sobre personas en diferentes hogares; cada hogar tiene varios sujetos en él. Si la variable `hhold` identifica el hogar en el que vive un individuo y `age` registra la edad de cada individuo, el siguiente ejemplo hallará la edad media de las personas más mayores en todos los hogares:

```

sort hhold age
by hhold : gen oldest = _n == _N
summ age if oldest

```

El siguiente comando hallaría la edad media del resto de los miembros del hogar: `summ age if ~oldest.`

Otro ejemplo: supongamos que tiene datos de la renta personal de los miembros de los hogares. Desea hallar la renta total del hogar, sumando entre todos los que componen cada hogar. Puede hacerlo con el siguiente conjunto de comandos:

```

sort hhold
by hhold : gen hhinc = sum(inc)
by hhold : replace hhinc = hhinc[_N]

```

Esta última operación se logra más fácilmente utilizando una de las funciones `egen`:

```

egen hhinc = sum(inc), by(hhold)

```

4 Resumen de comandos Stata

En esta sección ofrecemos una breve explicación de los comandos de Stata más importantes. Mediante el subrayado se indican las abreviaturas permitidas. No abuse de éstas en archivos con extensión `.do`, ya que dificulta la interpretación de los comandos por parte del programa.

4.1 Ayuda

Stata	Descripción
-------	-------------

<code>help help</code>	Ayuda interactiva acerca de la ayuda del sistema.
<code>help topic</code>	Ayuda interactiva sobre <i>topic</i> (comandos Stata), por ejemplo <code>help regress</code> .
<code>search string</code>	Enumeración de comandos Stata relativos al término <i>string</i> , por ejemplo <code>search regression</code> . También se incluyen artículos aparecidos en el boletín técnico de Stata y programas disponibles en archivos de programas Stata.
<code>lookfor string</code>	Lista las variables que contienen <i>string</i> en el nombre o etiqueta de la variable, por ejemplo <code>lookfor age</code>

4.2 Lectura y edición de archivos de datos

Stata	Descripción
<code>use</code>	Obtiene un archivo de sistema Stata para procesar.
<code>save filename</code>	Guarda como un archivo de sistema Stata; no olvide <code>replace</code> si desea sobrescribir un archivo existente.
<code>merge commonvar using filename</code>	Añade <i>variables</i> de otro archivo de sistema al archivo actual. Ambos archivos han de tener una variable común, <i>commonvar</i> , que identifique los casos a combinar. Ambos deben clasificarse sobre <i>commonvar</i> .
<code>append filename</code>	Añade <i>cases</i> de otro archivo de sistema.
<code>compress</code>	Intenta comprimir el archivo de datos convirtiendo, por ejemplo, <i>4 byte reals</i> a enteros, si ello es posible sin perder información. Este comando ahorra mucho espacio en el disco duro y en la memoria interna.
<code>edit</code>	Introduce datos directamente en el editor o cambia el valor de las variables existentes. (No disponible en la versión de línea de comandos de Stata).
<code>input varlist</code>	Introduce datos interactivamente. Teclee <code>end</code> para detener.
<code>infile varlist</code>	Lee datos ASCII sin formato o con formato.
<code>infix varlist</code>	Lee datos ASCII con formato fijo.
<code>insheet using filename</code>	Lee datos ASCII con formato delimitado con tab/coma.

<code>outfile using filename</code>	Edita archivos ASCII sin formato (opcionalmente con un diccionario).
<code>outsheet using filename</code>	Edita archivos ASCII delimitados con tabulador o comas.

4.3 Modificación interactiva de datos

Stata	Descripción
<code>generate newvar = expr</code>	Crea una nueva variable <i>newvar</i> utilizando una expresión <i>expr</i> .
<code>replace oldvar = expr</code>	Modifica los valores de una variable existente.
<code>edit / browse varlist</code>	Introduce datos directamente en el editor o cambia el valor de las variables existentes con una interfaz similar a una hoja de cálculo. (No disponible en la versión de línea de comandos de Stata).
<code>for</code>	Repite un comando para una lista de variables, una lista numérica o una lista de cadenas arbitrarias.
<code>egen</code>	Numerosas extensiones útiles para el comando <code>generate</code> , incluida la generación de medias “en filas”, etc.
<code>recode varname</code>	Recodifica la variable <i>varname</i>
<code>reshape</code>	Permite cambiar la organización de un conjunto de datos entre “ancho” y “largo”. Muy útil para gestionar conjuntos organizados individualmente (personas, lugares, cosas) en unidades de tiempo.
<code>collapse</code>	Agrega variables. Por ejemplo, si tenemos un conjunto de datos organizado por condados, puede agregar los datos por estados con el comando <code>collapse</code> . Atención: <code>collapse</code> sobrescribe valores existentes de variables.

4.4 Procedimientos descriptivos

Stata	Descripción
<code>describe</code>	Lista de nombres de variables, etiquetas, nº de observaciones, etc.

<code>ds</code>	Lista compacta de nombres de variables.
<code>summarize</code>	Estadísticas resumidas de observaciones válidas.
<code>summarize, detail</code>	Estadísticas detalladas, incluidos cuantiles, asimetría y curtosis.
<code>by varname : summ</code>	Estadísticas para subgrupos.
<code>tab varname, summ()</code>	Resumen para subgrupos.
<code>tabulate</code>	Tablas de una y dos variables categóricas.
<code>tab varname, plot</code>	Histogramas.
<code>by varname : tab</code>	Tablas multimodo.
<code>table</code>	Tablas multimodo, con formato mejorado.
<code>inspect</code>	Más resúmenes univariantes para inspección de datos.
<code>correlate</code>	Matriz de covarianza o de correlación de variables.
<code>pwcorr</code>	Matriz de correlación con un borrado por emparejamiento de casos faltantes.
<code>pcorr</code>	Correlaciones parciales.
<code>spearman</code>	Correlación por rangos de Spearman.
<code>count if exp</code>	Número de observaciones mantenidas por expresión <i>exp</i> .
<code>list</code>	Lista de observaciones.
<code>display expr</code>	Calculadora rudimentaria.

4.5 Gráficos

Stata	Descripción
<code>graph, box</code>	Diagramas de caja.
<code>graph, hist</code>	Histogramas (por defecto si sólo se da una variable).
<code>graph, oneway</code>	Diagramas de frecuencia tipo código de barras.
<code>graph, matrix</code>	Plot matricial de diagramas de dispersión bimodales.
<code>graph, twoway</code>	Diagrama de dispersión (por defecto si se dan dos o más variables).

<code>avplot</code>	Gráficos de variable añadida tras la regresión.
---------------------	---

4.6 Procedimientos estadísticos

Stata	Descripción
<code>regress</code>	Regresión
<code>ttest</code>	<i>t</i> -test

4.7 Procedimientos varios

Stata	Descripción
<code>sort</code>	Clasifica las observaciones de una o más variables.
<code>order</code>	Vuelve a clasificar las variables en la matriz de datos.
<code>rename</code>	Cambia del nombre de una variable.
<code>drop if expr</code>	Elimina las observaciones que satisfacen <i>expr</i>
<code>drop varlist</code>	Elimina las variables en <i>varlist</i>
<code>drop all</code>	Elimina todas las variables.
<code>clear</code>	Elimina todas las variables. Necesario antes de introducir nuevos datos.
<code>sample</code>	Muestra aleatoria de las observaciones a partir de los datos.
<code>keep</code>	Lo contrario de <code>drop</code>
<code>label</code>	Etiqueta las variables y los valores de las variables.
<code>do filename</code>	Ejecuta un archivo ASCII de comandos Stata. Asume que la extensión del archivo es <code>.do</code> , salvo que se especifique otra cosa.
<code>run filename</code>	Igual que <code>do</code> , pero ejecuta el archivo en modo <i>quietly</i> (sin salidas).
<code>exit</code>	Salir de Stata. (No se permite si se han modificado los datos y no se han guardado).
<code>exit, clear</code>	Salir de Stata incluso si no se han guardado los datos modificados.

<code>log using filename</code>	Realiza gran cantidad de entradas y salidas.
---------------------------------	--