

# **STATA**

## **Guía de utilización**

**MICROECONOMETRÍA**

**César Alonso Borrego**  
**Rocío Sánchez Mangas**

**Universidad Carlos III de Madrid**

# Índice

## 1. ASPECTOS GENERALES

- 1.1. Stata es “case sensitive”
- 1.2. Sintaxis de las órdenes
- 1.3. Recursos de memoria
- 1.4. Comando de ayuda
- 1.5. Precisión de las variables
- 1.6. Abreviaturas de comandos
- 1.7. Salir de Stata

## 2. GESTIÓN DE DATOS

- 2.1. Introducción de datos por teclado
- 2.2. Gestión de ficheros
- 2.3. Gestión de variables
- 2.4. Análisis descriptivo de datos

## 3. CREACIÓN DE VARIABLES

- 3.1. Comando *generate*
- 3.2. Extensiones
- 3.3. Reemplazar variables
- 3.4. Variables categóricas y variables indicador
- 3.5. Generación de retardos y diferencias

## 4. CREACIÓN DE GRÁFICOS

## 5. UNIÓN DE FICHEROS

## 6. FICHEROS DO

## 7. ESTIMACIÓN

- 7.1. Estimación por MCO
- 7.2. Estimación por VI
- 7.3. Estimación de modelos de elección binaria
- 7.4. Estimación de modelos censurados
- 7.5. Obtención de resultados tras la estimación
- 7.6. Contrastes de hipótesis
- 7.7. Estimación con datos de panel

# 1. Aspectos generales

Stata es un paquete estadístico diseñado para el análisis descriptivo de datos y la implementación de diferentes técnicas de estimación.

Trabajaremos con grandes bases de datos que contienen información de diferentes variables para un conjunto de individuos o empresas. En Stata, los ficheros de datos tienen extensión *.dta*. Supongamos que tenemos un fichero llamado *datos.dta* con seis variables llamadas *iden*, *year*, *var1*, *var2*, *var3* y *dum*, que contienen cierta información sobre una muestra de 30 individuos a lo largo de 5 años, es decir, en total, 150 observaciones. Supongamos que la variable *iden* identifica a los individuos y la variable *year* hace referencia al año, y que la información está organizada de la siguiente forma:

<i>iden</i>	<i>year</i>	<i>var1</i>	<i>var2</i>	<i>var3</i>	<i>dum</i>	<i>categ</i>
1	1991	45.6	6.2	65.2	1	5
1	1992	23.6	4.1	18.5	1	10
1	1993	12.5	3.5	65.1	0	10
1	1994	78.9	4.5	23.3	1	20
1	1995	45.3	5.4	14.7	0	5
2	1991	16.5	7.8	65.9	1	5
2	1992	56.9	6.2	87.2	1	10
2	1993	14.3	1.3	10.6	0	20
2	1994	12.5	3.6	11.7	1	20
2	1995	10.1	5.2	13.8	0	5
3	1991	12.3	2.5	14.3	1	5
3	1992	45.2	1.9	15.4	1	10
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
30	1994	26.5	9.8	85.3	0	20
30	1995	70.8	1.2	23.0	1	5

Estos datos nos servirán como ejemplo para ilustrar los comandos de Stata que permiten el análisis descriptivo y la estimación de modelos econométricos.

## 1.1. Stata es “case sensitive”

Stata distingue entre mayúsculas y minúsculas, de forma que las variables *var1* y *Var1* son distintas.

## 1.2. Sintaxis de las órdenes

En Stata la sintaxis de las órdenes es la siguiente:

**comando** *lista de variables* [**if** *expression*] [**in** *expression*], [*opciones*]

Todo lo que aparece entre corchetes es opcional. Con **in** podemos referirnos a un cierto rango de los datos y con **if** podemos introducir expresiones lógicas. Los operadores lógicos se escriben como:

&	y
	ó
~	negación

mientras que los operadores de relación vienen dados por:

>	mayor que
<	menor que
>=	mayor o igual que
<=	menor o igual
==	igual
!=	distinto(también puede escribirse: ~=)

### 1.3. Recursos de memoria

Por defecto, la memoria con la que trabajamos en un PC en Stata es de 1024K. Pero si la base de datos que vamos a manejar es muy grande, es posible que no tengamos memoria suficiente. Podemos incrementar los recursos de memoria con el comando **set memory**. Ejemplo:

**set memory 20000**

incrementa los recursos de memoria hasta 20000K.

### 1.4. Comando de ayuda

Mediante la instrucción

**help comando**

obtendremos la ayuda referente al comando que hemos escrito después de **help**.

### 1.5. Precisión de las variables

Las variables numéricas se almacenan en Stata según uno de los siguientes tipos: **byte**, **int**, **long**, **float** (por defecto) y **double**. Las variables de tipo **byte** son almacenadas en 1 byte, las de tipo **int** en 2 bytes, las de tipo **long** y **float** en 4 bytes y las de tipo **double** en 8 bytes. Las variables alfanuméricas, es decir, las que contienen texto, son de tipo cadena (**string**), que se denota por **str#** (str1, str2, ..., str80). En cuanto al formato de visualización, es decir, la forma en que los datos son presentados, es el siguiente: para variables numéricas, viene expresado como **%w.d**, seguido de uno de estos tres formatos: **e**, **f**, **g**. Con **w** denotamos un número entero que especifica la anchura del formato, mientras que **d** indica el número de dígitos que siguen al punto decimal. Para variables alfanuméricas, el formato es **%ws**, donde **s** indica **string** y **w** es un número entero que indica la anchura dada a la variable. Por defecto, el formato que asigna Stata a los distintos tipos de variable es:

<b>byte</b>	<code>%8.0g</code>
<b>int</b>	<code>%8.0g</code>
<b>long</b>	<code>%12.0g</code>
<b>float</b>	<code>%9.0g</code>
<b>double</b>	<code>%10.0g</code>
<b>str#</b>	<code>%#s</code> (ó <code>%9s</code> si la anchura es menor de 9)

Es posible cambiar el formato de visualización de las variables con el comando **format**. Por ejemplo, supongamos que en nuestros datos la variable *var1* es tipo **float**, formato `%9.0g`. Si queremos darle formato `%10.2g` la instrucción será:

```
format var1 %10.2g
```

El formato de visualización no afecta a la precisión de almacenamiento de los datos. La cantidad de memoria que consumen nuestros datos puede reducirse con el comando **compress**. Este comando reduce el tamaño de los datos del siguiente modo:

```
double pasa a long, int o byte
float pasa a int o byte
long pasa a int o byte
int pasa a byte
string pasa a longitud de cadena más corta
```

## 1.6. Abreviaturas de comandos

Stata permite referirnos a la mayoría de los comandos utilizando sólo sus tres primeras letras (incluso sólo la primera en algunos casos). Así, por ejemplo, el comando **generate** puede escribirse como **gen**, el comando **tabulate** como **tab**, etc. Hay algunas excepciones que deben escribirse sin abreviar, como **compress**. En las próximas secciones veremos estos y otros comandos.

## 1.7. Salir de Stata

Podemos terminar una sesión de Stata con el comando **exit**.

# 2. Gestión de datos

## 2.1. Introducción de datos por teclado

Con el comando **edit** accedemos al editor de Stata. Aparece una ventana a modo de hoja de cálculo donde podemos introducir datos.

## 2.2. Gestión de ficheros

- Cargar un fichero de datos en memoria. La extensión que Stata asigna a los ficheros de datos es *.dta*. Para cargar un fichero en memoria sólo tenemos que teclear el comando

correspondiente, **use**, seguido del nombre del fichero, sin necesidad de indicar su extensión.

**use** *datos.dta* (ó **use** *datos*)

- Descargar el fichero y dejar libre la memoria: **clear**

Mediante la instrucción

**use** *datos*, **clear**

cargamos en memoria el fichero *datos.dta*, descargando previamente cualquier fichero que estuviera en uso en ese momento.

- Grabar el fichero:

**save** *datos.dta*

Si el fichero ya existe y queremos reemplazarlo:

**save** *datos.dta*, **replace**

- Descripción del contenido del fichero de datos cargado en memoria:

**describe**

Este comando proporciona información sobre el número de observaciones y el número, nombre, tipo y formato de las variables del fichero de datos.

- Ordenación de datos:

Como podemos observar en la tabla de datos a la que venimos haciendo referencia, los datos están ordenados primero, según la variable *iden*, y después, según la variable *year*. El comando para ordenar los datos es **sort**. Así,

**sort** *year*

ordena los datos en orden ascendente según los valores de la variable *year*.

**sort** *year iden*

los ordena según la variable *year* en primer lugar y después según la variable *iden*. Algunos comandos de Stata, como **egen** (lo veremos más adelante) desordenan los datos, de forma que es preciso asegurarse siempre de cómo los tenemos ordenados y ordenarlos de la forma más conveniente. Si los datos están ordenados respecto a alguna variable, el comando **describe**, tras listar el nombre de todas las variables y su etiqueta (si la tienen) nos indica cómo están ordenados los datos.

## 2.3. Gestión de variables

- Borrar y mantener variables en un fichero:

Ejemplos:

**drop** *var1*

borra la variable *var1*

**drop** *var1 var2*

segunda borra las variables *var1* y *var2*

**keep** *iden year var1*

mantiene las variables *iden*, *year* y *var1*, borrando el resto de las variables del fichero. Stata admite plantillas para referirse a una lista de variables. Por ejemplo,

**drop** *var\**

borra todas las variables que empiezan por *var*.

Los comandos **drop** y **keep** también permiten borrar y mantener, respectivamente, un rango concreto de observaciones. Así, por ejemplo,

**drop if** *year>1992*

borra, en todas las variables del fichero en uso, todas las observaciones para las que *year>1992*

**drop in** *5/15*

borra, en todas las variables, todas las observaciones desde la 5 hasta la 15.

**keep if** *dum==1 & var1 >20*

mantiene, en todas las variables, todas las observaciones para las que *dum=1* y *var1 >20*, borrando el resto de observaciones.

- Renombrar variables: supongamos que queremos renombrar la variable *var1* de forma que pase a llamarse *nvar*. Entonces la orden será:

**rename** *var1 nvar*

- Poner una etiqueta a una variable: Supongamos que queremos poner a la variable *iden* la etiqueta *identificador de individuo*, para tener siempre presente qué significa esa variable. La orden será:

**label variable** *iden "identificador de individuo"*

Si hemos etiquetado una variable, el comando **describe** nos dará, además del nombre, la etiqueta de la variable.

## 2.4. Análisis descriptivo de datos

- Contar el número de observaciones de nuestro fichero de datos: **count**

Para contar el número de observaciones que cumplen cierta condición:

```
count if iden>20 & dum==1
```

nos da el número de observaciones del fichero en uso que cumplen la condición señalada detrás de **if**.

- Mostrar los datos de una o varias variables: el comando **list** muestra el valor de cada una de las observaciones de las variables contenidas en el fichero en uso. También podemos especificar qué variable(s) queremos listar, o incluso qué rango de observaciones en esas variables. Ejemplos:

```
list var1 var2  
list var1 in 2/10  
list var1 in 90/l  
list year if iden>10 & year!=1990  
list year var1 in 2/50 if var1 <= var2 / dum==0
```

donde con la orden **in** 2/10 señalamos el rango que queremos listar: de la segunda observación a la décima de la variable especificada, en este caso *var1*. Con **in** 90/l el rango señalado es desde la observación 90 hasta la última (*last*). Con la orden **if** *expresión* sólo listamos, para las variables especificadas, aquellas observaciones que cumplen la expresión escrita después de **if**.

- Información más detallada de una o varias variables:

```
inspect var1 var2
```

indica el número de valores positivos, negativos, cero y faltantes (missing values), así como el número de valores enteros y no enteros de las variables *var1* y *var2*. Los “missing values” se señalan en Stata mediante un punto (.). Se considera que un “missing value” es mayor que cualquier valor.

- Estadísticos descriptivos de una variable:

```
summarize var3 (o bien, sum var3)
```

proporciona información acerca del número de observación, la media, la desviación típica, el mínimo y el máximo de la variable especificada.

```
summarize var3 , detail (o bien, sum var3,d)
```

proporciona, además, coeficientes de asimetría y curtosis y varios percentiles de la variable especificada.



Si queremos información acerca de todas las variables del fichero en uso, escribiremos simplemente **sum**; o, si queremos información más detallada, **sum,d**.

También podemos crear nombres de listas de variables para referirnos a ellas de forma conjunta y evitar así tener que escribir cada una de ellas cada vez que vayamos a utilizarlas. Supongamos que queremos incluir en dicha lista variables *var2*, *var3* y *var4*. Entonces escribimos:

```
global grupo "var2 var3 var4"
```

Una vez creado el grupo, si queremos información de dichas variables mediante el comando **sum**, basta escribir:

```
sum $grupo
```

Stata también permite obtener medias ponderadas. Las ponderaciones se expresan mediante el comando **weight**. Por ejemplo, si la variable de ponderación es *pond*, la instrucción

```
sum var1 [weight=pond]
```

proporciona información sobre la media y desviación típica de la variable *var1* ponderada según la variable *pond*. Stata permite diferentes tipos de ponderaciones.

- Tablas de frecuencias:

Comando **tabulate**, o abreviado, **tab**:

```
tabulate var1
```

muestra la tabla de frecuencias de la variable *var1*

```
tabulate var1 var2
```

muestra una tabla de frecuencias de doble entrada, con información cruzada de las variables *var1* y *var2*;

```
tabulate var3 if var1 + var2 > 30 & year != 1991
```

muestra la tabla de frecuencias de la variable *var3* sólo para las observaciones que cumplen la expresión especificada después de **if**.

- Tablas de estadísticos descriptivos:

```
table year, contents (mean var1 sd var1 )
```

muestra una tabla con los diferentes valores de la variable *year*. Con las opciones señaladas en **contents** obtendremos, en este caso, el valor medio y la desviación típica de la variable *var1* para las observaciones correspondientes a cada una de los valores de la variable *year*.

```
table year dum, contents(n var1 mean var2 ) row
table year dum if year<1993, contents (median var3 ) row col
```

añadiendo además la opción **row** (**col**), la tabla incluirá una fila (columna) adicional con los valores totales, para cada valor de *year* y *dum*, de las opciones que aparecen en **contents**. Todo lo que aparece después del comando **table** y las variables especificadas son opciones, y por tanto, no son necesarias en la sintaxis del comando. Si no aparecen, es decir, si sólo escribimos

```
table year
table year dum
```

obtendremos sólo las tablas de frecuencias correspondientes, como con el comando **tabulate**.

Las opciones más frecuentes dentro de **contents** son:

```
n      (número de observaciones)
mean  (media)
sd    (desviación típica)
median (mediana)
max   (máximo)
min   (mínimo)
p1    (primer percentil)
p2    (segundo percentil)
...
p98   (percentil 98)
p99   (percentil 99)
iqr   (rango intercuartílico)
```

### 3. Creación de variables

#### 3.1. Comando generate

El comando para generar variables es **generate**, que puede abreviarse como **gen**  
Ejemplos:

```
generate var4 = var1 + var2
generate var5 = var1 / var2 if dum==1
```

En la segunda de estas órdenes, Stata asignará un “missing value”,., a aquellas observaciones que no cumplen la condición señalada después de **if**.

También podemos generar una variable considerando cada valor de una variable de referencia. Para ello es necesario que los datos estén ordenados según dicha variable.  
Ejemplo:

```
by iden: gen var4 = sum(var2 )
```

Esta orden trata las observaciones por grupos según el valor de la variable especificada después de **by**. Dado que en nuestro ejemplo la variable *iden* toma 30 valores diferentes (1,2,...,30), tenemos 30 grupos de observaciones. La instrucción anterior genera una variable *var4* que va asignando a las observaciones de cada grupo, la suma de las observaciones de *var2* en ese grupo. Es decir:

<i>iden</i>	<i>year</i>	<i>var2</i>	<i>var4</i>
1	1991	6.2	6.2
1	1992	4.1	10.3
1	1993	3.5	13.8
1	1994	4.5	18.3
1	1995	5.4	5.4
2	1991	7.8	13.2
2	1992	6.2	19.4
2	1993	1.3	20.7
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Siempre que generamos variables por grupos (con el comando **by**, como en el ejemplo anterior), Stata va mostrando por pantalla mensajes intermedios que pueden ser suprimidos si al inicio de la instrucción escribimos el comando **quietly**. Es decir, la instrucción anterior podría escribirse como:

**quietly by iden: gen var4 =sum(var2 )**

Además, podemos opcionalmente especificar la precisión de la variable que vamos a generar. Por ejemplo:

**gen double var5 =ln(var1 )**

Es importante tener en cuenta el tipo de variable que vamos a generar si queremos determinar una precisión distinta a la que Stata asigna por defecto. Así, si vamos a generar una variable que tiene decimales y la generamos como **byte**, tendremos errores. Por ejemplo, 12.8 será almacenado como 12.

Con las órdenes que utilizan **by** combinado con **generate**, también es posible utilizar los comandos **if** e **in** para especificar un subconjunto de observaciones.

### 3.2. Extensiones

Existe una extensión del comando **generate**, el comando **egen**. Este comando genera variables que vienen expresadas como ciertas funciones de otras variables. También es posible declarar la precisión de la variable que queremos generar, así como combinar el comando con **if** e **in**. Ejemplos:

**egen var7=max(var1 +var2 ) if dum==1**

genera una variable que para todas las observaciones para las que *dum=1*, vale el valor máximo de la suma de *var1* y *var2* . Para el resto de observaciones, las que no cumplen la condición en **if**, Stata asigna un missing value.

**egen double** *var8*=sd(*var1* ), **by**(*iden*)

genera una variable de tipo **double**, que contiene para cada observación de cada uno de los 30 grupos definidos por cada valor de *iden*, la desviación típica de los valores de *var1* en ese grupo.

Son muchas las funciones que podemos utilizar para generar variables con el comando **egen**, entre ellas: count, iqr, max, mean, median, min, sd, sum, etc.

Es importante tener en cuenta que el comando **egen** desordena los datos. Por ello es necesario volver a ordenarlos después mediante el comando **sort**.

Cuando se usa la función “sum” los comandos **gen** y **egen** pueden confundirse. Veamos un ejemplo que clarifica la diferencia existente entre ellos. Consideremos que nuestros datos son sólo las 5 primeras observaciones de nuestra tabla inicial. Supongamos las siguientes instrucciones:

**gen** *var8*=sum(*var1* )  
**egen** *var9*=sum(*var1* )

El resultado será (omitimos el resto de variables):

<i>var1</i>	<i>Var8</i>	<i>var9</i>
45.6	45.6	205.9
23.6	69.2	205.9
12.5	81.7	205.9
78.9	160.6	205.9
45.3	205.9	205.9

### 3.3. Reemplazar variables

Comando **replace**: funciona igual que el comando **generate**, pero en lugar de crear nuevas variables, reemplaza los valores de variables ya existentes. Ejemplos:

**replace** *var1* =1.5 **in** 1/10  
**replace byte** *dum*=2 **if** *dum*==0

Cuando usamos el comando **replace**, Stata nos informa acerca del número de valores que han sido modificados.

### 3.4. Variables categóricas y variables indicador

Una variable indicador indica si para la muestra estudiada, cierta característica es cierta o falsa. Podemos pensar, por ejemplo, que la variable *dum* recoge los resultados de una

encuesta en la que se preguntó a los individuos si habían viajado al extranjero en el período considerado, siendo las respuestas posibles: 1 si el individuo ha viajado al extranjero y 0 en caso contrario. Una variable categórica divide a la muestra en grupos dependiendo de cierta característica. Podemos pensar, por ejemplo, que la variable *categ* recoge los resultados de una encuesta en la que se preguntó a los individuos sobre el número de conciertos a los que asistieron en el período considerado, siendo las respuestas posibles: 5 si fueron a menos de 5 conciertos, 10 si presenciaron entre 5 y 15 conciertos y 20 si fueron a más de 15. Toda variable indicador es una variable categórica, pero lo contrario no es cierto. Stata permite convertir variables continuas en variables categóricas o variables indicador y convertir variables categóricas en variables indicador.

- Construcción de una variable indicador a partir de una variable continua

```
gen d1=(var1 >25)
```

genera una variable indicador *d1*, que tomará el valor 1 para las observaciones para las que *var1*>25 y 0 para el resto. Por defecto, la variable *d1* que genera Stata será **float**, pero dado que sólo va a tomar los valores 1 y 0, es mejor generarla como **byte** para ahorrar memoria. Por tanto, es mejor generarla como:

```
gen byte d1=(var1 >25)
```

Hay que tener en cuenta que si existen missing values, éstos son considerados mayores que cualquier otro valor, por tanto, es mejor generar la variable de forma que sólo tome los valores 0 y 1 para las observaciones donde no hay missing value. Entonces, la forma de generar la variable es:

```
gen byte d1=(var1 >25) if var1 != .
```

De esta forma, la variable *d1* tendrá un missing value en aquellas observaciones en las que la variable *var1* = .

Otra forma de generar la variable *d1* es en dos pasos, del siguiente modo:

```
gen d1=1 if var1 >25 & var1 != .  
replace d1=0 if var1 <25
```

La primera de estas instrucciones asigna 1 a las observaciones que cumplen la condición señalada en **if** y missing values al resto. La segunda reemplaza por 0 los valores que cumplen la condición señalada en **if**, sin alterar el resto.

- Construcción de una variable categórica a partir de una variable continua.

Supongamos que queremos generar una variable *d2* cuyos valores estén en función de los valores de *var1* de la siguiente forma: *d2* vale 0 si *var1*≤15, vale 1 si *var1*∈(15,30] y vale 2 si *var1*>30. Como antes, para ahorrar recursos de memoria, estamos interesados en generar la variable tipo **byte**. La forma de generar la variable categórica *d2* es la siguiente:

```
gen byte d2=0 if var1 <=15
```

```
replace d2=1 if var1 >15 & var1 <=30
replace d2=2 if var1 >30 & var1 !=.
```

- Construcción de variables indicador a partir de variables categóricas

Supongamos que queremos crear, para nuestra variable categórica *categ*, que toma los valores 5, 10 y 20, variables indicador llamadas, por ejemplo, *indi*, que indiquen cuándo la variable *categ* toma cada uno de esos valores. Podemos hacerlo con la siguiente instrucción:

```
tab categ, gen(indi)
```

El resultado será una tabla de frecuencias de la variable *categ*, y la creación de tres nuevas variables: *indi1* (que vale 1 cuando *categ*=5 y 0 en el resto de los casos), *indi2* (que vale 1 cuando *categ*=10 y 0 en el resto de los casos) e *indi3* (que vale 1 cuando *categ*=20 y 0 en el resto de los casos).

### 3.5. Generación de retardos y diferencias

Es necesario que los datos estén ordenados adecuadamente. Supongamos que tenemos un fichero de datos de series temporales que contiene dos variables: *time*, que recoge el período de tiempo, y *serie*, que recoge los valores de una serie temporal, por ejemplo, el IPC. Supongamos que queremos generar una variable, por ejemplo *serie\_1*, que recoja los valores de la variable *serie* desfasados un período. Las instrucciones serán:

```
sort time
gen serie_1=serie[_n-1]
```

En general, si queremos generar retardos de orden k, escribiremos

```
gen serie_k=serie[_n-k]
```

y si queremos generar las primeras diferencias de la variable serie (la llamamos, por ejemplo, *d1serie*), la instrucción será

```
gen d1serie=serie-serie[_n-1]
```

Para generar diferencias de orden k,

```
gen dkserie=serie-serie[_n-k]
```

Los datos que venimos manejando no forman series temporales sino un panel de datos. En este caso, no tiene sentido crear los retardos y las diferencias como en el ejemplo anterior. Lo más razonable será crear retardos y diferencias en las series temporales que tenemos para cada individuo. Por ejemplo, para crear la variable *var1* retardada un período escribiremos

```
sort iden year
quietly by iden: gen var1_1=var[_n-1]
```

o bien,

```
gen var1_1=var[_n-1] if iden==iden[_n-1]
```

### Generación de retardos y diferencias de forma automática

Stata también permite crear retardos y diferencias de forma automática a través de los operadores **L** y **D**. Antes de utilizar estos comandos, es necesario especificar qué variable identifica el tiempo, si trabajamos con series temporales; si trabajamos con datos de panel, hay que especificar además la variable que identifica al individuo. El comando que permite identificar los datos como series temporales o como datos de panel es **tsset**. Por ejemplo:

```
tsset iden year
```

identifica los datos como datos de panel, siendo *iden* la variable que identifica al individuo y *year* la que identifica el tiempo. Una vez utilizado el comando **tsset**, la generación de retardos y diferencias es muy simple. Ejemplos:

```
gen var_1=L1.var1
```

genera la variable *var1* retardada un período, y la almacena en una variable que, en este ejemplo, hemos llamado *var1\_1*.

```
gen var1_2=L2.var1
```

genera la variable *var1* retardada dos períodos y la almacena con el nombre *var1\_2*.

```
gen d1var1=D1.var1
```

genera la variable *var1* en primeras diferencias, y la almacena con el nombre *d1var1*.

```
gen d2var1=D2.var1
```

genera la variable *var1* en segundas diferencias y la almacena con el nombre *d2var1*.

Si después de los operadores **L** y **D** no especificamos ningún número, Stata genera por defecto primeros retardos y primeras diferencias respectivamente.

Además Stata permite, mediante los comandos **L** y **D**, utilizar retardos y diferencias de variables aunque éstas no hayan sido generadas previamente. Por ejemplo,

```
gen x=var1+L1.var2
```

genera la suma de la variable *var1* y el primer retardo de la variable *var2*.

## 4. Creación de gráficos

Los gráficos se generan en Stata con el comando **graph**. Si sólo se hace referencia a una variable y no se especifica nada acerca del tipo de gráfico, Stata genera un histograma de la variable considerada. Si queremos otro tipo de gráfico, hay que especificarlo como una de las opciones del comando **graph**. Si especificamos dos variables, Stata genera un diagrama en el que aparece la primera variable en el eje de ordenadas y la segunda en el eje de abscisas. Si especificamos más de dos variables, la última será representada en el eje de abscisas y el resto en el eje de ordenadas. Con el comando **graph** también es posible utilizar **by**, **in** e **if**.

Ejemplos:

```
graph var1
graph var1 in 2/10, box
graph var1 var2 if dum==1
by iden: graph var1
```

Para grabar un gráfico cuando lo creamos, debemos incluir la opción **saving**, y un nombre para el fichero que contendrá el gráfico. La extensión que Stata asigna a este fichero es *.gph*. Ejemplo:

```
graph var1 var2 , saving(graf1,replace)
```

genera un gráfico de la variable *var1* sobre la variable *var2* y lo guarda en un fichero llamado *graf1.gph*. La opción **replace** reemplaza este fichero en caso de que ya exista. Si queremos recuperar por pantalla un gráfico ya creado, lo haremos llamando al fichero en el que lo hemos guardado. Por ejemplo:

```
graph using graf1
```

Hay muchas opciones que pueden especificarse en un gráfico de Stata referentes al tipo de gráfico, la forma de conectar y señalar los puntos en un gráfico, los títulos del gráfico y los ejes, etc.

## 5. Unión de ficheros

La fusión de ficheros de datos en Stata se hace con los comandos **append** y **merge**. Supongamos que además de nuestro fichero *datos.dta* tenemos un fichero *datos2.dta*, que contiene información sobre otras variables referentes a la misma muestra de individuos del fichero *datos.dta*. Las variables que aparecen en el fichero *datos2.dta* serán *iden* y *year* (que identifican las observaciones como en el fichero *datos.dta*) y otras variables, por ejemplo, *x1* y *x2*. Si queremos construir un fichero llamado *total.dta* que contenga toda la información de los anteriores, los comandos serán:

```
use datos.dta
```



```
merge iden year using datos2.dta  
save total.dta
```

Los ficheros que se fusionan deben estar ordenados respecto a la lista de variables que se utilizan para identificar las observaciones, en este caso, *iden* y *year*. Con el comando **merge**, además de añadir al fichero *datos.dta* las variables de *datos2.dta*, Stata crea una variable llamada *\_merge*, que toma el valor 1 para observaciones que están en el primer fichero pero no en el segundo, 2 para las observaciones que están en el segundo fichero pero no en el primero y 3 para las que están en los dos ficheros. Tabularemos esta variable, y nos quedaremos sólo con las observaciones para las que *\_merge*=3. Es decir:

```
tab _merge  
keep if _merge==3  
save total.dta, replace
```

Si en lugar de unir ficheros que contienen distintas variables para las mismas observaciones queremos unir dos ficheros que contienen las mismas variables, pero distintas observaciones, el comando a utilizar es **append**.

## 6. Ficheros do

Hay dos modos de trabajar en Stata:

1. de modo interactivo: escribiendo instrucciones en la línea de comandos y viendo el resultado por pantalla sin guardarlo.
2. en modo “batch”, mediante ficheros *.do*. Esta es la forma óptima de trabajar.

Un fichero *.do* es un fichero de texto ASCII que contiene un conjunto de comandos e instrucciones de Stata que serán ejecutados con el comando **do** seguido del nombre del fichero. Podemos crear un fichero *.do* con el editor de Stata o con cualquier editor de texto. Supongamos que hemos creado un fichero llamado *fich.do*. Para ejecutarlo, escribimos en la línea de comandos de Stata la siguiente instrucción:

```
do fich
```

Veamos un ejemplo. Supongamos que *fich.do* contiene las siguientes instrucciones:

```
use datos.dta  
describe  
list var1 var2 in 1/10  
gen var1 2=var1 *var2  
save datos.dta, replace
```

Cuando tecleamos la instrucción **do fich**, obtendremos por pantalla el resultado de los comandos e instrucciones que aparecen en el fichero *fich.do*.

Sin embargo, la forma idónea de trabajar es crear un fichero de salida, que tendrá extensión *.log*, en el que se almacenarán los resultados. Consideremos un fichero *fich2.do* que contiene las siguientes instrucciones:

```
#delimit;  
set more off;  
capture log close;  
log using salidas, replace;  
** esto es un comentario**;  
clear;  
use datos.dta;  
gen var12=var1 *var2 ;  
describe;  
list iden year var12 if dum==1;  
save datos.dta, replace;  
log close;
```

Veamos cómo funciona cada una de las instrucciones de este fichero *fich2.do*. Hay que tener que es posible que alguna de las instrucciones del fichero ocupe más de una línea. En este caso, habrá que especificar si un salto de línea es el fin de una instrucción o no lo es. La primera instrucción de este fichero especifica como delimitador el símbolo “;”. De esta forma, sólo si Stata encuentra dicho símbolo, lo entenderá como fin de una instrucción e inicio de la siguiente. La segunda instrucción (**set more off**) indica que los resultados serán mostrados por pantalla hasta el final. Es decir, si los resultados ocupan más de una pantalla, la instrucción **set more off** impide que los resultados vayan apareciendo poco a poco. Esto es lo habitual y lo óptimo si vamos a llevar los resultados a un fichero de salida. Mediante la instrucción

**log using *salidas*,replace**

Stata abre un fichero llamado *salidas.log* donde se almacenarán los resultados. Si el fichero ya existe, será reemplazado. Si queremos escribir comentarios, debemos iniciarlos con el símbolo “\*”. A continuación aparecen una serie de instrucciones que queremos ejecutar. Finalmente, la instrucción

**log close**

cierra el fichero de salida que hemos abierto. La instrucción

**capture log close**

evita errores en el caso de que no haya podido ejecutarse un fichero *.do* hasta el final. Supongamos que no escribimos esa instrucción en el fichero *.do* que queremos ejecutar y que en dicho fichero hemos abierto un fichero de salida *.log* mediante **log using**. Supongamos que alguna de las instrucciones que aparecen después de **log using** contiene algún error (por ejemplo, un comando mal escrito). Entonces, Stata detecta el error y no sigue ejecutando el resto de instrucciones del fichero. Cuando subsanemos el error y volvamos a ejecutar el fichero *.do*, Stata nos dará un nuevo mensaje de error en el que nos dice que el fichero *.log* que pretendemos abrir con la orden **log using** ya está

abierto. Para evitar este problema, la instrucción **capture log close** indica que aún en el caso de no poder ejecutar un fichero *.do* hasta el final, si hemos abierto un fichero *.log*, Stata debe “capturar” la instrucción **log close** y cerrar el fichero *.log*. Para ejecutar el fichero, tecleamos **do fich2** en la línea de comandos. Para ver los resultados, abrimos el fichero *salidas.log* con cualquier editor o procesador de texto o directamente en Stata.

## 7. Estimación

Todas las instrucciones de estimación de Stata tienen la misma sintaxis y comparten la mayoría de las opciones.

### 7.1. Estimación por MCO

El comando es **regress** seguido de la variable dependiente y la lista de variables independientes, separadas por un espacio. Ejemplo:

```
regress var1 var2 var3
```

El comando **regress** permite hacer estimación para un cierto rango de observaciones (con **in** e **if**), así como estimación por grupos de observaciones (con **by**). En este caso, los datos deben estar ordenados con respecto a la variable que define los grupos. Ejemplos:

```
regress var1 var2 var3 if categ==2
```

```
sort iden  
by iden: regress var1 var2 var3
```

Para obtener la matriz de varianzas-covarianzas de los estimadores, el comando es **vce**. Añadiendo a este comando la opción **corr**, es decir, mediante la instrucción

```
vce, corr
```

obtenemos la matriz de correlación de los estimadores.

Para hacer los resultados de la estimación robustos a heterocedasticidad se utiliza la opción **robust**. Ejemplo:

```
regress var1 var2 var3 , robust
```

También es posible hacer una estimación ponderada, dando mayor o menor peso a las observaciones según su importancia en una variable de referencia. Supongamos que la variable de referencia es *var4*. Entonces haremos la estimación ponderada mediante la instrucción:

```
regress var1 var2 var3 [weight=var4]
```

## 7.2. Estimación por variables instrumentales

Supongamos que queremos regresar *var1* sobre *var2* y *var3*, y disponemos de dos variables instrumentales llamadas *ins1* e *ins2*. La estimación VI se hará incluyendo entre paréntesis la lista de instrumentos después de los regresores:

```
regress var1 var2 var3 (ins1 ins2)
```

Para simplificar las órdenes y si se van a hacer regresiones sucesivas con las mismas variables explicativas, pueden crearse nombres de listas de variables. Supongamos que tenemos los regresores *var2*, *var3*, *var4*, *var5* y *var6* y queremos hacer regresiones sobre el grupo *var2*, *var3*, *var4*. Entonces:

```
global grupo "var2 var3 var4"  
regress var1 $grupo
```

Si en lista de regresores están incluidos *var2*, *var3*, *var4*, *var5* y *var6*, podemos simplificar la orden mediante:

```
regress var1 var2 -var6
```

El número de variables que pueden incluirse en una estimación son, por defecto, 40. Si queremos ampliar el número de variables, el comando es **set matsize**. Ejemplo:

```
set matsize 150
```

aumenta hasta 150 el número posible de variables a incluir en la estimación. El número máximo de variables permitido es 800.

## 7.3. Estimación de modelos de elección binaria

Existen varios comandos que permiten estimar modelos de elección binaria, dependiendo del supuesto que se realice sobre la función de distribución que genera la probabilidad de escoger uno u otro suceso. Bajo el supuesto de normalidad, tendremos el modelo **probit**, y el comando utilizado lleva este mismo nombre, con sintaxis análoga a la de la regresión lineal. Sea *y* la variable dependiente (binaria) y *var1*, *var2* y *dum* los regresores. Entonces la instrucción para la estimación es:

```
probit y var1 var2 dum
```

Si se quieren medir directamente los efectos sobre la probabilidad del suceso de referencia (aquel para el que  $y=1$ ) de cambios infinitesimales en cada una de las variables explicativas (o de un cambio discreto en variables explicativas discretas), puede aplicarse directamente el comando **dprobit**, con idéntica sintaxis:

```
dprobit y var1 var2 dum
```

Suponiendo que la distribución es logística, tendremos el modelo logit, y el comando lleva este mismo nombre, con idéntico formato:

**logit** y *var1 var2 dum*

A diferencia del modelo probit, para calcular los efectos de las variables sobre la probabilidad no existe un comando similar a **dprobit**, teniendo que hacer los cálculos en Stata.

#### 7.4. Estimación de modelos censurados

Existen varios comandos que permiten estimar distintos modelos cuando la variable dependiente presenta censura o truncamiento. El más general es **cnreg**, con la sintaxis:

**cnreg** y *var1 var2, censored(varname)*

donde la opción **censored(varname)** es obligatoria: requiere especificar una variable que establece si para cada observación existe o no censura. Dicha variable puede tomar tres valores: 0 si la observación no está censurada, -1 si presenta censura inferior, 1 si presenta censura superior. En el caso de modelo tobit con censura inferior, esta variable tomaría valores 0 ó 1.

Para el caso particular del modelo Tobit, podemos usar el comando

**tobit** y *var 1 var2, ll(#) ul(#)*

donde **ll(#)** y **ul(#)** especifican los puntos de censura inferior y superior respectivamente: observaciones con  $y \leq ll()$  presentan censura inferior; aquellas con  $y \geq ul()$  presentan censura superior y las restantes con  $ll() < y < ul()$  no están censuradas. No es necesario especificar los puntos de truncamiento y censura, de manera que bastaría con poner **ll**, **ul**, o ambos. En el caso de un modelo tobit con censura inferior, pueden omitirse **ul**, **ll**, y Stata considerará que el punto de censura es el mínimo observado en los datos (por ejemplo, si observamos ceros y valores positivos, entenderá que la censura se produce para las observaciones con valor igual a 0).

En el modelo generalizado de selección, además de emplear un procedimiento bietápico en el que se realiza una estimación probit auxiliar primero y se calcula la inversa del ratio de Mills para estimar luego el modelo con la muestra truncada y la inversa del ratio de Mills como variable explicativa adicional, también puede estimarse el modelo generalizado de selección por máxima verosimilitud bajo normalidad mediante el comando **heckman**. Para ello, hay que definir primero las ecuaciones de selección y del modelo para la variable latente. Por ejemplo:

**eq modelo:** *y var1 var2 ...*

**eq probit:** *d z1 z2 ...*

**heckman** *modelo probit*

El comando **heckman** supone que el primer nombre (modelo) es el de la ecuación de la variable latente y el segundo (**probit**) contiene la lista de variables que determina el

truncamiento. Este comando estima por máxima verosimilitud ambas ecuaciones así como el coeficiente de correlación entre los errores de ambas ecuaciones ( $\rho$ ) y la desviación típica de la ecuación de la variable latente ( $\sigma$ ), así como el producto de ambas ( $\lambda = \rho\sigma$ ).

Es posible obtener el estimador en dos etapas en vez del máximo verosímil con este comando, para ello, debe escribirse

**heckman** *modelo probit*, **iterate(0)**

## 7.5. Obtención de resultados tras la estimación

El comando **predict** permite obtener el valor estimado de la variable dependiente, siempre referido a la última regresión estimada. Supongamos que queremos llamar *estvar1* al valor estimado de la variable dependiente. Entonces, la instrucción será:

**predict** *estvar1*

Hay varias opciones que pueden añadirse al comando **predict**. Ejemplos:

**predict** *sdpred*, **stdp**

calcula la desviación estándar de las predicciones y la almacena en una variable que hemos llamado *sdpred*.

**predict** *residuo*, **residuals**

calcula los residuos y los almacena en la variable *residuo*.

**predict** *sdresid*, **stdr**

calcula la desviación estándar de los residuos y los guarda en *sdresid*.

Si el modelo estimado ha sido un probit o un logit, existen dos predicciones de interés. La más importante es la de las probabilidades ajustadas, que se obtiene con la opción por defecto. Por ejemplo,

**predict** *predic*, **p**

almacena las predicciones en la variable *predic* (la opción **p** puede omitirse). El error estándar de las predicciones se obtiene con la opción **stdp**. Si se desea obtener el valor  $x_i'b$  (donde  $b$  son los estimadores de los parámetros del modelo) y almacenarlo en una variable *varname*, la instrucción correspondiente es:

**predict** *varname*, **xb**

Las opciones **p** y **xb** son muy útiles para obtener la estimación de la inversa del ratio de Mills para cada observación. El denominador es directamente proporcional a la

probabilidad del suceso para el que  $y=1$ . El numerador se calcula evaluando la densidad de la normal para los valores  $x_i$  de cada observación, lo que puede hacerse mediante el comando **egen** en la forma:

**egen numerad=normd(varname)**

donde *numerad* es un nombre de variable en que queremos almacenar el cálculo de dicha densidad y *varname* es el nombre de la variable en que hemos almacenado los valores  $x_i$  previamente generados.

## 7.6. Contrastes de hipótesis

### Contrastes de hipótesis lineales

El comando para contratar hipótesis lineales es **test**. Ejemplos:

**test var2**

contrasta la significación de la variable *var2*.

**test var2 var3**

contrasta la significación conjunta de *var2* y *var3*.

**test var2 +var3 =4**

contrasta la hipótesis  $var3 + var4 = 0$ .

**testparm var2 - var5**

contrasta la significación conjunta del grupo de variables *var2*, *var3*, *var4* y *var5*.

**test var2 =2**

**test var3 +var4 =0, accumulate**

contrasta la hipótesis  $var3 + var4 = 0$ , conjuntamente con la hipótesis  $var2 = 2$ . Es decir, la opción **accumulate** permite contrastar una hipótesis conjuntamente con la anterior hipótesis contrastada.

### Contrastes de hipótesis no lineales

El comando a utilizar es **testnl**. Primero definimos con el comando **eq** la ecuación que queremos contrastar y luego aplicamos el comando **testnl** sobre esa ecuación. Para referirnos a un parámetro que acompaña a una variable, por ejemplo, *var2*, la sintaxis es *\_b[var2]*. Ejemplos:

**eq hip1: \_b[var2] \* \_b[var3] =5**  
**testnl hip1**

### Contraste de Hausman

Stata permite realizar el contraste de Hausman utilizando el comando del mismo nombre. Para ello, hay dos alternativas:

- a) obtener primero el estimador menos eficiente; utilizar el comando **hausman, save**; después obtener el estimador más eficiente y utilizar el comando **hausman**.

Ejemplo:

```
reg y var1 var2 var3 (ins1 var2 var3)
hausman, save
reg y var1 var2 var3
hausman
```

- b) obtener primero el estimador más eficiente; utilizar el comando **hausman, save**; después obtener el estimador menos eficiente y utilizar el comando **hausman, less**.

Ejemplo:

```
reg y var1 var2 var3
hausman, save
reg y var1 var2 var3 (ins1 var2 var3)
hausman, less
```

### 7.7. Estimación con datos de panel

Todos los comandos que empiezan por **xt** permiten realizar operaciones relativas a datos de panel (**xtdata**, **xtgls**, **xtreg**, etc.).

Para utilizar estos comandos es preciso que una variable identifique al individuo (familia, país, empresa, etc.) y otra al período temporal a que corresponde cada observación.

El comando **xtreg** permite obtener diversos estimadores de panel. Supongamos que queremos estimar una ecuación de *var1* sobre *var2* y *var3*.

```
xtreg var1 var2 var3 , be i(iden)
```

calcula el estimador intra-grupos (opción **be**: between groups). La variable en **i(.)** es la que identifica al individuo.

```
xtreg var1 var2 var3 , fe i(iden)
```

calcula el estimador de efectos fijos (opción **fe**: fixed effects). Si no se especifica ninguna de las dos opciones anteriores, el estimador que se obtiene por defecto es el de efectos aleatorios (opción **re**: random effects)

```
xtreg var1 var2 var3 , re i(iden)
```



Después de esta estimación puede hacerse un contraste de Breusch y Pagan de que los efectos individuales no son significativos. El comando es:

**xttest0**

También puede realizarse un contraste de que los efectos individuales no están correlacionados con los regresores mediante un contraste de Hausman. El comando es:

**xthaus**