Introducción a Stata

Econometría I [1]

Universidad Nacional de La Plata

Marzo, 2020

[1] Basado en presentación de Javier Alejo del año 2016

Contenido

- Introducción a Stata
  - Interface
  - Sintaxis
  - Comandos básicos
  - Abreviaturas
- Organización de un proyecto en archivos “DO” y “LOG”
- Gestión de base de datos (dataset)
  - Tipo de datos y conversión de tipos
  - Importación y exportación de datos
- Gráficos
- Ejercicios

¿Qué es Stata?

- Stata es un sistema que permite la gestión de base de datos y la realización de cálculos estadísticos y econométricos.
- La gran ventaja de Stata es que se basa en un lenguaje de programación que respeta una sintaxis.
- Quienes sepan principios de programación podrán asimilar a Stata como un sistema de programación de alto nivel con algunos aspectos similares a Pascal, C, o Basic.
- Existen versiones de Stata para Windows, Linux y Mac.

¿Dónde se aprende?

- En este curso: solo lo necesario para Econometría I.
- Stata ofrece varias alternativas:
  - Para una primera aproximación es suficiente con el "User's Guide".
  - Manuales detallados por comandos.
  - Stata Press: libros sobre temas especificos (estadística, econometría, demografía, etc.)
  - Stata Journal: articulos sobre nuevos métodos, comandos y otros tópicos de programación.
- En la web: blogs, videos tutoriales, etc.

Interface

- La interface de Stata comprende el entorno de trabajo. En una primera aproximación se trabajará con la interface de ventanas, para más adelante experimentar con la interface mediante archivos do y log.
- Las diferentes ventanas que conforman la interface son:
  - Review: muestra el historial de comandos recientemente utilizados.
  - Variables: expone las variables que comprenden el dataset actualmente en memoria.
  - Results: Muestra los resultados obtenidos de la aplicación de los comandos.
  - Command: es para introducir comandos mediante el teclado.

Sintaxis y comandos básicos

- Stata trabaja mediante la especificación por parte del usuario de órdenes que se denominan comandos.
- Los comandos conforman un lenguaje de comunicación con el programa, por lo que existe una determinada sintaxis que tiene la siguiente estructura general:

```
[by varlist:]  comando [varlist] [=exp] [if exp] [in range] [,
opciones]
```

- Los corchetes indican elementos opcionales. De hecho existen comandos que comprenden sólo una palabra.
- Veremos diferentes ejemplos de comandos simples que usan distintas partes de la estructura de la sintaxis

Bases de datos (dataset)

- Utilizaremos un dataset de ejemplo denominado **basewdi.dta** (obtener del material de clases en el sitio web, y guardar en **C:\Intro a stata**).
- Este dataset contiene datos extraidos de la World Development Indicators para 27 países para los años 2000 y 2005.
- El comando para cargar el dataset en la memoria de Stata es:

```
use C:\Intro a stata\basewdi.dta
```

- Requisitos:
  - No debe haber un dataset previo en memoria.
  - Previamente habrá que decirle a Stata en que carpeta se está trabajando (comando **cd C:\Intro a stata**).

```
[by varlist:]  comando [varlist] [=exp] [if exp] [in range] [, opciones]
```

- Si la carga del dataset fue exitosa, veremos que las ventanas Review, Variables y Stata Results se modificaron.
- Un ejemplo de comando que funciona solo invocando su nombre es el comando **describe**, que muestra una descripción de la base de datos y la lista de variables que contiene:

```
describe
```

- Otros casos son los comandos **browse**, que muestra toda la base de datos, y **summarize**, que muestra estadisticas descriptivas:

```
browse
summarize
```

```
[by varlist:]  comando [varlist] [=exp] [if exp] [in range] [, opciones]
```

- Se utiliza para hacer referenca a una o más variables. Por ejemplo, estadísticas descriptivas de una variable:

```
summarize pob
```

- Para un grupo de variables:

```
summarize pob pbi expo
```

- Lista de variables (en este caso todas las variables que están entre pob y impo)

```
summarize pob-impo
```

- Variables que empiecen con la letra p

```
summarize p*
```

`[by varlist:] comando [varlist] [=exp] [if exp] [in range] [, opciones]`

- Se utiliza cuando se quiere restringir la aplicación del comando a observaciones que cumplen con ciertas restricciones.
- Para ello se utiliza el "si condicional" (**if** en inglés).
- Por ejemplo: descripción estadística de la variable pbi de aquellos países cuya población es mayor a 250 mil habitantes.

```
summarize pbi if...
```

- Operadores de comparación:
  - *Igual*: ==
  - *Distinto*: !=
  - *Mayor (menor)*: > ; (<)
  - *Mayor o igual (menor o igual)*; >= (<=)

[by varlist:] comando [varlist] [=exp] [if exp] [in range] [, opciones]

- Se utiliza cuando se quiere restringir la aplicación del comando a observaciones que cumplen con ciertas restricciones.
- Para ello se utiliza el "si condicional" (**if** en inglés).
- Por ejemplo: descripción estadística de la variable pbi de aquellos países cuya población es mayor a 250 mil habitantes.

```
summarize pbi if pob > 250000
```

- Operadores de comparación:
  - *Igual*: ==
  - *Distinto*: !=
  - *Mayor (menor)*: > ; (<)
  - *Mayor o igual (menor o igual)*; >= (<=)

- Operadores lógicos:
  - *And*: &
  - *Or*: |
- Operador jerárquico: el paréntesis que determina el orden de aplicación de las sentencias condicionales.
- Algunos ejemplos más complejos de sentencias condicionales son los siguientes:

```
summarize pbi if (pob > 250000) & (consumo < 6000)
summarize pbi if (pob > 250000 & consumo < 6000) | anio==2000
summarize pbi if (pob > 250000) & (consumo < 6000 | anio==2000)
summarize pbi if !(pob > 250000) & (consumo < 6000 | anio==2000)
```

```
[by varlist:]  comando [varlist] [=exp] [if exp] [in range] [, opciones]
```

- Permite aplicar el comando a un rango de observaciones, de acuerdo al orden del dataset.
- Ejemplo 1: descripción estadística de las 10 primeras observaciones del dataset

```
summarize pbi in 1/10
```

- Ejemplo 2: aplicar el comando a las últimas 10 observaciones.

```
summarize pbi in -10/-1
```

- El componente **[in range]** depende del ordenamiento del dataset.
- Los comandos para ordenar un dataset son **sort** y **gsort**.
- El comando **sort** permite ordenar sólo de manera ascendente de acuerdo a la variable que se especifica:

```
sort pob
```

- El comando **gsort** permite ordenar en cualquier sentido.
- De manera descendente según la población de cada país:

```
gsort -pob
```

- De manera ascendente:

```
gsort +pob
```

`[by varlist:] comando [varlist] [=exp] [if exp] [in range] [, opciones]`

- Se utiliza generalmente con sólo dos comandos: **generate** y **replace**.

```
generate nueva = 0
```

- Permite crear una nueva variable en el dataset. Es requisito indicar la definición de dicha variable nueva.
- En este caso la variable se llama nueva y tiene valor 0 en todas las observaciones.
- También puede crearse una variable nueva mediante operaciones algebraicas basadas en otras variables.

- Ejemplo 1: crear la variable que se llame pob2 que contiene la población en millones

```
generate pob2 = pob/1000000
```

- Ejemplo 2: crear una variable que contenga el PBI per cápita

```
generate pbipc = pbi/pob2
```

- Ejemplo 3: crear una variable con el saldo de la Cuenta Corriente

```
generate ctacte = expo - impo
```

- Ejemplo 4: crear una variable que contenga el índice de apertura

```
generate apertura = 100 * (expo + impo ) / pbi
```

- Para poder verificar los valores de pob y pob2 puede aplicarse el comando **browse**:

```
browse pob pob2
```

- O alternativamente

```
browse po*
```

- También puede crearse una variable con valores nulos (en Stata se indica ".").

```
generate nulo = .
```

- Otro comando que usa **[=exp]** es **replace** que permite reemplazar valores de una variable ya creada.

```
replace pob2 = 0 if pob < 250000
```

- A las variables ya generadas se les puede crear una etiqueta (label), como se puede ver en la ventana de variables o haciendo un **describe**:

```
label var mercosur "=1 si el pais es miembro del Mercosur"
```

- También se puede etiquetar los valores de una variable, especialmente si la variable es categórica. Se procede en dos pasos
  - Paso 1: Creo las etiquetas de los valores

```
label define etiqueta 0 "No Mercosur" 1 "Mercosur"
```

  - Paso 2: le aplico la etiqueta definida anterioremente a los valores de la variable (o las variables) que correponda

```
label values mercosur etiqueta
```

  - Las variables con valores etiquetados aparecen en color azul al hacer **browse**

`[by varlist:] comando [varlist] [=exp] [if exp] [in range] [, opciones]`

- Permite sistematizar la aplicación del comando por grupos de observaciones.
- Los grupos están definidos por los distintos valores de la variable indicada en varlist.
- Es requisito ordenar el dataset por la variable que se va a usar en el **[by varlist:]**

```
sort region
```

- Luego:

```
by region:  summarize apertura
```

bysort

- Lo mismo se podría haber obtenido haciendo:

```
summarize apertura if region=="America Central"
summarize apertura if region=="America del Norte"
summarize apertura if region=="America del Sur"
```

pero debe notarse que este segundo método es muy engorroso si la variable que agrupa los datos tiene más de dos opciones.

- Una opción que permite la misma funcionalidad pero en una sola linea de código es **bysort**:

```
bysort region:  summarize apertura
```

- Con esto no es necesario ordenar previamente al dataset por la variable deseada.

`[by varlist:] comando [varlist] [=exp] [if exp] [in range] [, opciones]`

- Existen comandos que aceptan opciones adicionales.
- Éstas son especificadas en la sintaxis luego de una coma.
- Por ejemplo: resumen estadístico más detallado

```
summarize pbipc, detail
```

- De esta manera, el comando **summarize** ahora brinda una descripción estadística distinta de la que hace por defecto.

```
use C:\Intro a stata\basewdi.dta, clear
```

- Con esta opción evitamos la precaución de no tener un dataset previo en la memoria al momento de abrir una base, mediante el comando **use.**

Otros comandos básicos

- Para tabular cantidades de observaciones según distintos valores de una variable, tenemos el comando **tabulate**.
- Por ejemplo:

```
tabulate region
```

- Permite conocer cuantas observaciones hay con los diferentes valores que tiene la variable region.
- Este comando **tabulate** puede ser combinado con el **summarize**, de la siguiente manera:

```
tabulate region, summarize(apertura)
```

- De esta manera se obtiene la media, el desvío estándar y la frecuencia, según los distintos valores de la variable region.

- También puede usarse **tabulate** con dos variables.

```
tabulate region anio
```

- Permite conocer cuantas observaciones hay con las diferentes combinaciones de valores entre las variables pais y region.
- Obviamente, cualquiera de estos comandos acepta la introducción de condiciones lógicas, como por ejemplo:

```
tabulate pais anio if expo==.
```

- El comando **tabstat** es más avanzado para obtener la descripción estadística de variables.
- El uso es el siguiente:

```
tabstat pbipc if anio==2005, statistics(mean)
```

- Obtenemos la media de la variable pbipc
- Pero también podemos obtener muchos otros estadísticos descriptivos, como por ejemplo

```
tabstat pbipc if anio==2005, statistics(mean sum N max min range)
```

- . . . y de más de una variable

```
tabstat pbipc apertura if anio==2005, statistics(mean sum max min
range)
```

Abreviaturas

- Todos los comandos y variables usados pueden ser abreviados.
- La regla es que la abreviatura puede realizarse siempre que no se confunda con otro comando.
- Existen algunas excepciones a esta regla:
  - Los comandos "destructivos" no se abrevian: **drop**, **clear**
  - Existe el comando **describe** que se abrevia con d, a pesar de confundirse con otros (es un comando muy utilizado).
- Ejemplos de abreviatura:

```
tab reg, sum(ape)
```

Archivos "DO" y "LOG"

- Hasta ahora la interacción con Stata ha sido mediante el tipeo de comandos en la ventana "Stata commands".
- Archivo "DO" (do-file): son archivos de texto que contienen una secuencia de comandos.
- Al ejecutar dicha secuancia, los resultados serán visualizados en Stata, pero no guardados.
- Una forma de guardar esos resultados es utilizando un archivo LOG.
- Archivos "LOG" (log-file): son archivos de texto en donde se almacena una copia de todo lo visualizado en la ventana Results de Stata.

Proyecto

- Está compuesto por todos los archivos que intervienen en nuestra interacción con Stata.
- En resumen, un proyecto simple contiene los siguientes archivos:

Archivo	¿Qué hace?
dta	contiene los datos necesarios
do	ejecuta una secuencia de comandos
log	guarda los resultados

- Esta estructura de proyecto resulta muy útil para la resolución de gran parte de los TPs.

Archivos DO

- Un archivo DO es de tipo "texto plano" (sin formatos).
- La idea central es que contenga una secuencia de comandos que nos permita obtener ciertos resultados.
- Para la creación de un DO-FILE tenemos dos alternativas:
  1. Editor de textos que tiene incluido Stata.
  2. Editor de textos externo.

Editor de textos incluido en Stata

Se puede abrir desde el Menú ó mediante el siguiente comando:

```
doedit
```

Editor de textos externo

- Hay varias opciones disponibles:
  - Notepad++: `http://notepad-plus-plus.org/`
  - Editplus: `http://www.editplus.com/`
  - Textpad: `http://www.textpad.com/`
  - Crimson: `http://www.crimsoneditor.com/`
- Cada uno requiere instalar un archivo para que reconozca la sintaxis de Stata.

Mi primer archivo DO

- Utilizando el dataset basewdi.dta escribiremos nuestro primer archivo DO mediante el editor de textos. Para ello realizamos las siguientes acciones:
- Copiamos en **C:\Intro a stata** el archivo **basewdi.dta**. Esta será nuestra carpeta de trabajo.
- En el editor de Stata escribimos

```
cd "C:\Intro a stata"
use basewdi.dta, clear
describe
summarize pbi expo impo
```

- Guardamos el archivo con el nombre **wdi.do** en la carpeta **C:\Intro a stata**
- En Stata command ejecutamos do-file wdi. Hay dos maneras:
  1. Utilizando el botón "Execute (do)"
  2. Mediante el comando do, escribiendo en la ventana de comandos: **do wdi.do**
- Si todo funcionó bien, habremos ejecutado nuestro primer archivo DO y en la ventana Stata Result estará el resultado de los comandos describe y summarize.
- Nota: pueden incorporarse comentarios dentro del archivo DO de la siguiente manera: /* Este es un comentario */. También es un comentario una línea iniciado con asterisco *

Mi primer archivo LOG

- Los resultados que se registran en la ventana Stata result pueden ser almacenados en un archivo de texto de extensión .log El código a agregar para obtener un archivo LOG es el siguiente:

```
cd "C:\Intro a stata"
use basewdi.dta, clear
capture log close
log using wdi.log, text replace
describe
summarize pbi expo impo
log close
```

Funcionamiento:

- **log using** hace que se empiecen a registrar los resultados en el archivo wdi.log mientras que "log close" los cierra.
- **replace** implica que en cada nueva ejecución del programa los resultados se sobrescriben.
- Solo quedan registrados los resultados entre el **log using** y el siguiente **log close**.

Comentarios:

- Existen otras formas de exportar resultados (en forma de tablas, texto y gráficos).
- El archivo LOG es una de las más primitivas pero la más simple (y por lo tanto útil para principiantes).

Estructura del dataset

- El contenedor de datos en Stata se denomina DATASET.
- Es una tabla de doble entrada donde las columnas se denominan variables y las filas observaciones.

Observación	Variable1	Variable2	Variable3
1	Dato 1,1	Dato 2,1	Dato 3,1
2	Dato 1,2	Dato 2,2	Dato 3,2
...	...	...	...
N	Dato 1,N	Dato 2,N	Dato 3,N

- En cada celda se pueden guardar datos de diferentes tipos.
- El comando count reporta el total de observaciones:

```
count
```

Tipos de datos

- Tipos de datos en Stata: (i) números, (ii) palabras y (iii) fechas.
- Datos numéricos: admite varios formatos

Nombre	Tipo de números	Límite inferior	Límite superior
Byte	Enteros	-127	100
Integer	Enteros	-32,767	32,740
Long	Enteros	-2,147,483,647	2,147,483,620
Float	Con decimales	-1.70141173319*10^38	1.70141173319*10^38
Double	Con decimales	-8.9884656743*10^307	8.9884656743*10^307

- Datos en palabras: se pueden almacenar palabras con hasta un máximo de 2045 caracteres.
- Fechas: son números con un formato especial de visualización.

- Para conocer el tipo de datos de todas las variables que componen el dataset se utiliza:

```
describe
```

- Si se quiere saber el tipo de dato de una variable específica se utiliza:

```
describe pob
```

- Para crear una variable numérica especificando el tipo de dato que queremos, se utiliza:

```
generate byte cantidad = 0
generate float descuento = 0
```

- Para crear una variable que contenga palabras se utiliza el siguiente comando:

```
generate str8 zona = ""
```

- El límite máximo es de 2045 caracteres en Stata 13 MP.
- En este caso, la variable de tipo string especifica la cantidad máxima de caracteres a almacenar.

```
replace zona = "Mercosur" if pais=="arg" | pais=="bra" | pais=="par" | pais=="ury" |

pais=="ven"
```

- Posible contenido de un string:
  - *Datos identificatorios*: esta información no puede ser utilizada directamente en el análisis estadístico.
  - *Valores no categóricos*: se trata de variables con números en formato string. Para esto es util el comando **destring**.

Importación de datos

- Stata puede importar datos con otros formatos que no son los de Stata.
- Para Stata 12 o más antiguas, el comando que permite esto es **insheet**.
- Los formatos genéricos usualmente tienen estas extensiones:
  1. txt: son archivos de texto plano. Por lo general usa tabulaciones como separador de variables.
  2. csv: similares a los anteriores pero el separador de variables es un "punto y coma" (;).

Importación de datos

- A partir de Stata 13 el comando genérico para esta tarea es **import**.
- Por ejemplo, en el caso delimitado por comas:

```
import delimited using dataset1.csv, clear
```

- Pero también permite importar archivos Excel:

```
import excel using dataset2.xlsx, clear
```

Exportación del dataset

- El dataset que está en la memoria de Stata puede ser exportado a un archivo **txt**, **csv** o de Excel.
- En Stata 12 (o anteriores), el comando para realizar esta tarea es **outsheet**.
- En Stata 13 se usa el comando **export**:

```
export delimited archivo1, replace
```

- Y en el caso de exportar a un Excel:

```
export excel using archivo2.xslx, replace
```

- El comando **export** excel permite exportar el contenido del dataset a un archivo excel

```
preserve
table pais, c(mean pob mean pbi) replace
export excel using “tabla1.xlsx", replace
restore
```

- El comando **preserve** toma una fotografía de la base de datos.
- El comando **restore** devuelve la base de datos tal como estaba antes del preserve.

Gráficos

- Gráfico de dispersión/nube de puntos

```
twoway scatter pbi consumo
```

- Gráfico de dispersion/nube de puntos condicionando las observaciones que aparecen en el gráfico (ej: consumo<200000)

```
twoway scatter pbi consumo if consumo<200000
```

Gráficos

- Gráfico de dispersión junto a un gráfico de la recta de regresión estimada por MCO (PBI predicho)

```
twoway (scatter pbi consumo if consumo<200000) (lfit pbi consumo if
consumo<200000)
```

- Gráfico de la relación lineal entre pbi y consumo para los años diferentes de la base

```
twoway (line pbi consumo if consumo<200000 & anio==2000) (line pbi
consumo if consumo<200000 & anio==2005)
```

Ejercicios

Se pide:

- ¿Cuál es el país con mayor población en 2005?
- Dentro de cada región, ¿cuál es el país con mayor pbi?
- ¿Qué paises pertenecen al Mercosur ? ¿Qué promedio de expo e impo tiene cada uno de ellos?
- ¿Cuál es la población promedio de Mercosur en 2005? ¿Y la total?
- Generar una variable string que clasifique paises segun su lengua sea español, inglés, portugués y otros.