Notas de clase – Taller de Stata[1]

Clase 2 – Variables I

Contenido

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

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

Taller de Stata
Clase 2 – Variables I
Miguel Garzón Ramírez, Cristhian Acosta Pardo,
Facultad de Economía, Universidad de los Andes

1. Comandos para la descripción de variables

Los comandos en *Stata* funcionan con base en información utilizada en forma de escalares, vectores, matrices y macros. En las primeras clases estudiaremos el funcionamiento de comandos básicos para explorar bases de datos y variables. En clases futuras estudiaremos la información utilizada en la ejecución de los comandos y programaremos tareas con ella. Por ahora nos conformaremos con indicar la información que exponen los comandos.

En esta clase utilizaremos los comandos summarize, tabulate, tabstat, describe e inspect, que permiten explorar el contenido bases de datos y variables en *Stata.*

1.1. Summarize

El comando sum presenta estadísticas descriptivas para las variables. Sin especificar ninguna opción adicional presenta el número de observaciones, la media, la desviación estándar, el mínimo y el máximo. Al especificar la opción *detail* (o su abreviación d) se presenta, además de lo anterior, los rangos de algunos percentiles, la varianza, la curtosis y la asimetría o sesgo (skewness).

1.2. Tabulate

El comando tabulate (tab) presenta una tabla de frecuencias de una variable determinada. Los resultados del comando tabulate incluyen el porcentaje de observaciones en cada valor que toma la variable especificada y, en el caso en que los valores tengan etiquetas, el resultado de *tab* presenta estas etiquetas. Por ejemplo, para ver la distribución de carros importados y nacionales, corremos el siguiente comando:

Ej. 1 ⟶ tab *n_def*

El comando tab está configurado de manera que no tiene en cuenta los valores faltantes. Sin embargo, si se quiere incluir estos en la tabla de frecuencias sólo es necesario especificar la opción *miss.*

Ej. 2 ⟶ tab *n_def*, miss

Para omitir las etiquetas de los valores existe la opción *nolabel.*

Ej. 3 ⟶ tab *genero*, miss nolabel

Dado que el comando tab no hace estimaciones, es un comando tipo *r*. Es posible cruzar información de dos variables utilizando el comando tab, a esto lo llamamos tabla de

contingencia. Por ejemplo, si queremos ver cómo se distribuye el número de reparaciones que han tenido los carros hasta el año 1.978 de acuerdo con el origen del automóvil, ejecutamos el siguiente comando:

Ej. 4 ⟶ tab *genero col_mixto*

Si se quiere ver la distribución de frecuencias de varias variables en el mismo comando, se debe utilizar el comando tab1*:*

Ej. 5 ⟶ tab1 *genero parcial1 parcial2*

1.3. Tabstat

El comando tabstat sirve para presentar una serie de estadísticas de diferentes variables en una sola tabla. Las estadísticas disponibles están en el archivo de ayuda del comando tabstat. La sintaxis del comando es la siguiente: tabstat *[lista de variables], stat(estadísticas a presentar)*

Si no se especifica nada, tabstat presenta únicamente la media de las variables especificadas. Ejemplo:

Ej. 6 ⟶ tabstat *parcial1 parcial2 n_def*

En la base de datos de ejemplo de *Stata* (⟶ webuse auto, clear), si queremos saber el promedio y la desviación estándar de la variable *price* y *rep78* de acuerdo si el carro es importado o fue fabricado al interior del país, el comando sería el siguiente:

Ej. 7 ⟶ tabstat *parcial1 parcial2 n_def, stat(mean sd) by(genero)*

Note que se especifica la opción *by* que es la que indica, por medio de la variable al interior del paréntesis el nivel al cual se calculan las estadísticas.

1.4. Describe

El comando describe genera un resumen de la base de datos en la memoria de *Stata*. En particular, lista las variables que componen la base, el tipo de almacenamiento, el formato, las etiquetas de valores y de variables.

Ej. 8 ⟶ describe *parcial1 parcial2 n_def*

1.5. Inspect

El comando inspect indica el número de valores diferentes que toma una variable y, así mismo, presenta una aproximación a un histograma directamente en la pantalla de resultados. Indica, del total de observaciones que hay, el total de valores faltantes, positivos, negativos, enteros y no enteros que toma la variable. El comando se puede especificar de tal forma que haga el inspect de varias variables al mismo tiempo:

Ej. 9 ⟶ inspect *parcial1 parcial2 n_def*

2. Generación de variables

Para hablar de las diferentes características que tienen las variables en *Stata* es fundamental saber cómo crearlas y modificarlas. Con el comando generate se pueden crear variables del tipo numéricas como se presenta a continuación:

Ej. 10 ⟶ gen *bogota*=0

El comando empleado en este ejemplo crea una variable que va a tomar el valor de 0 en todas las observaciones de la base de datos. Posterior a la creación de la variable se pueden realizar cambios en la misma por medio del comando replace. Estos reemplazos pueden realizarse para todas las observaciones al no incluir ninguna condición específica, o para un subconjunto particular de observaciones que cumplan con una condición tal y como se presenta en el Ej. 11.

Ej. 11 ⟶ replace *bogota*=1 if *depmuni*=="11001"

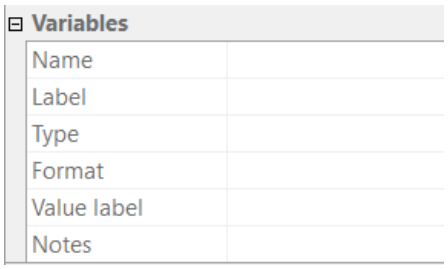
Vale la pena notar que siempre que se emplee un condicional de igualdad (e.g. if var1==#) debe emplearse el doble igual, ya que se está realizando una operación lógica. Esto es un requisito esencial de sintaxis de *Stata*. Para la creación de las variables de texto, los comandos a emplear son los mismos y su sintaxis no cambia en gran medida. La única diferencia radica en que para identificar un texto se deben emplear las comillas, como en la condición empleada en el Ej. 2.

Las operaciones básicas que se pueden utilizar en *Stata* con el comando *gen* se muestran en la Tabla 1. Todas las variables en *Stata* tienen cinco diferentes características: (i) nombre, (ii) etiqueta, (iii) tipo, (iv) formato, y (v) notas adjuntas. Adicionalmente, las variables de cualquier tipo numérico pueden tener una característica extra que es la presencia de (vi) etiquetas de valores. Estas características se pueden observar en la parte superior de la ventana de propiedades.

Tabla 1: Operaciones que puede ejecutar el comando *generate*

Símbolo	Operación
+,-	**Suma, Resta**
***,/**	**Producto, División**
^	**Potencia**
abs(x)	**Valor absoluto**
exp(x)	**Exponencial**
ln(x),log(x)	**Logaritmo natural**
log10(x)	**Logaritmo base 10**
sqrt(x)	**Raíz cuadrada**

Figura 1: Parte superior de la ventana de propiedades – Características de las variables



A continuación, se explicará de manera detallada alguna una de estas características y varios comandos asociados a estas.

3. Nombres

Los nombres de las variables son una manera en la que el usuario hace referencia a una información particular sobre un grupo de observaciones (e.g. género o edad de las personas). Estos pueden tener una longitud máxima de 32 caracteres, que únicamente pueden incluir letras (bien sea en mayúscula o en minúscula), números[2] y el símbolo raya al piso ("_"). A partir de Stata 14 los nombres de las variables pueden incluir tildes, aunque esto no es recomendable.

Al igual que los comandos, los nombres también pueden abreviarse, siempre y cuando no haya ambigüedad en la abreviación. Por ejemplo, en la base de datos "Base_Clase_2.dta" se puede ver algunas estadísticas descriptivas de la variable *ingresos_jefe_hogar* usando el comando summarize[3].

[2] Los números no pueden ser incluidos al inicio de los nombres.
[3] Si no conoce la función de este comando, en la parte final de estas notas se puede encontrar una descripción concisa.

Ej. 12 →[4] sum *ingresos_jefe_hogar*

Vale la pena notar que se introdujo el comando mediante su abreviación, la cual puede consultarse en el *help file* del comando. Se le puede dar a *Stata* la misma orden recién mencionada mediante el siguiente comando:

Ej. 13 → sum *ingresos*

Esto se puede realizar debido a que no hay ambigüedad en la abreviación; si se tratase de hacer esto empleando una abreviación que pueda referirse a más de una variable, tal y como se presenta en el Ejemplo 14, el software arrojará un mensaje de error.

Ej. 14 → sum *E*

En los casos en los que se quiera hacer referencia a una lista de variables puede emplearse el signo de cierre de interrogación (?), el símbolo asterisco (*), o el guion (-). El signo de cierre de interrogación se emplea para referirse a un carácter desconocido o variable en el nombre de las variables. Por otro lado, el símbolo asterisco se emplea para referirse a uno o más caracteres desconocidos o variables en el nombre de las variables. Finalmente, el guion se emplea para pedirle a *Stata* que tome todas las variables que se encuentran entre un par de variables. Para mayor claridad sobre estas abreviaciones se puede observar la Tabla 2.

Tabla 2. Ejemplos de abreviaciones empleando la base de datos de la clase

Comando	Variables que se referencian
sum *ED?*	Todas las variables que empiecen por ED y tengan un tercer carácter: EDA y EDI.
sum **A*	Todas las variables que terminen con la letra A: EDA e IRA.
sum *E**	Todas las variables que inicien con la letra E: EDA y EDI.
sum **A**	Todas las variables que contengan la letra A: EDA, IRA y MALT
sum *EDA-EDI*	Todas las variables que se encuentren entre EDA y EDI, incluyendo estas.

Ya que se mencionó el uso del guion para las abreviaciones, vale la pena resaltar que el comando order permite ordenar la base de datos, indicándole a *Stata* sí una variable debe ir al inicio, al

[4] La flecha indica la sintaxis del comando a introducir en la ventana de comandos de *Stata*

final o en una posición particular de la lista de variables[5]. Adicionalmente, se puede emplear este comando con la opción alphabetic para organizar la base en orden alfabético.

Adicionalmente, rename es el comando utilizado para renombrar variables, en su sintaxis se especifica en primer lugar el nombre actual de la variable y después el nuevo nombre de la variable. También se puede utilizar para renombrar varias variables al tiempo por medio de listas.

Ej. 15.1⟶ rename *EDA eda*

Ej. 15.2 ⟶ rename (*IRA MALT EDI) (ira malt edi)*

4. Tipos

La información de las bases de datos siempre se presenta o en (i) forma numérica o (ii) en forma de texto. De acuerdo a la forma y tamaño de la información, esta se puede guardar en diferentes variables de diferentes tipos. En la Tabla 3, se presentan los diferentes tipos de variables existentes para información en forma numérica.

Tabla 3. Tipos de variables numéricas[6]

Tipos	Número más pequeño	Número más grande	Número más cercano a 0 sin ser 0	Memoria (bytes)	Decimales
byte	-127	100	+/-1	1	No
int	-32,767	32740	+/-1	2	No
long	-2,147,483,647	2,147,483,620	+/-1	4	No
float	$-1.7014*10^{38}$	$1.7014*10^{38}$	$+/-10^{-38}$	4	Sí
double	$-8.9884*10^{307}$	$8.9884*10^{307}$	$+/-10^{-323}$	8	Sí

En cuanto a la información de texto, los tipos de las variables resultan más simples. Toda variable que contenga información de texto será tipo ***strL***, donde L corresponde al número máximo de caracteres que contiene alguna de las observaciones de dicha variable. La expresión *str* es la

[5] La sintaxis por defecto de este comando está diseñada para ubicar variables al inicio de la lista. Para trabajar con alguna posición diferente deben emplearse las opciones que se presentan en el *help file*.
[6] Tomada de: https://www.stata.com/manuals13/ddatatypes.pdf

abreviación de la palabra *string* del inglés, que se usa para hacer referencia a las variables de texto. El máximo número de caracteres que *Stata* permite almacenar es 2,045[7].

Para cambiar una variable de tipo se puede emplear el comando recast. Esto puede ser útil cuando se está empleando demasiada memoria en variables de manera innecesaria. Un ejemplo de esto es la variable EDA, de la base de datos de esta clase. Esta variable está almacenada como *float,* pero podría guardarse como una variable tipo *byte* dado que no tiene decimales y únicamente toma valores de cero (0) o uno (1).

Ej. 16 ⟶ recast byte *EDA*

Adicionalmente, para ahorrar la mayor cantidad de memoria posible se puede emplear el comando compress. Este comando convierte pasa todas las variables al tipo de variable que utiliza la menor cantidad de memoria sin perder información.

Vale la pena aclarar que si se tiene una variable que toma valores inferiores a 0.1, la aproximación numérica que hace Stata no es perfecta. Esto se debe a que los números decimales por defecto están almacenados con su representación binaria y hay números que no tienen una representación exacta en el lenguaje binario. Entre estos, el número 1/10 no tiene una representación exacta en binarios. Para ilustrar el punto anterior se presenta el siguiente ejemplo.

Ej. 17: La variable *pipps*, es un indicador de desarrollo cognitivo de los niños que puede tomar cuatro valores: 0.05, 0.1, 0.15 o 0.2. Se considera que un niño está teniendo problemas de desarrollo cognitivo si tiene un puntaje menor o igual a 0.1. Si queremos identificar cuántos niños se encuentran en esta delicada situación podemos emplear dos aproximaciones:

(1) ⟶ tab *pipps,* miss

(2) ⟶ count if *pipps*<=0.1

El primer comando muestra una tabla de frecuencias de los valores de la variable *pipps*. La opción miss le indica a *Stata* que reporte no solo a frecuencia de los valores sino también la frecuencia de los valores perdidos o *missing values*. Viendo la tabla resultante podemos notar que los niños que cumplen la condición que buscamos son 2,500.

[7] Este límite es para la versión 14 de Stata. Para la versión 13 el número máximo de caracteres es de 244. Esta es una de las razones por las cuales puede haber problemas al pasar bases de datos de una versión a otra del software.

Por otro lado, el segundo comando cuenta el número de observaciones para las cuales la variable *pipps* toma un valor menor o igual a 0.1. Con esta aproximación diríamos que los niños que cumplen la condición son 1,250, lo cual es contrario al número que ya habíamos observado. El problema está en que *Stata* no tuvo en cuenta a los niños que tienen un puntaje exactamente igual a 0.1. Para solucionar este problema, y no encontrar ninguna inconsistencia, se le debe especificar a *Stata* que tome valores menores o iguales a 0.1 en la representación tipo *float* de 0.1.

(2') ⟶ count if *pipps*<=float(0.1)

De esta manera podemos ver que las dos aproximaciones arrojan el mismo número, siempre y cuando se encuentren bien especificadas.

5. Formatos

Existen diferentes formatos en los que se puede presentar la información, de acuerdo a las necesidades del usuario. Es importante resaltar que cuando se habla del formato de las variables se hace referencia únicamente a como están son presentadas y no a la información que se encuentra almacenada. En la Tabla 4 se presentan los formatos numéricos más empleados.

Tabla 4. Formatos numéricos más empleados

Sintaxis formato	Nombre	Descripción	Ejemplo
%#$_1$.#$_2$ g	General	#$_1$ es el ancho total del número, y #$_2$ es el número de dígitos que se visualizan. Si el número es muy grande o muy pequeño, el formato se cambia automáticamente a %#$_1$.#$_2$ e.	%9.0g
%#$_1$.#$_2$ f	Fijo	#$_1$ es el ancho total del número, y #$_2$ es el número de decimales.	%9.2f
%#$_1$.#$_2$ e	Exponencial	Notación exponencial. #$_1$ indica el ancho total y #$_2$ indica el número de dígitos después del punto.	%10.7e
%t(x)	Fecha/Hora[8]	Notación temporal. La x puede ser c, C, d, w, q, h, y o g dependiendo de las necesidades	%td

Los formatos general y fijo permiten utilizar coma (,) como separador de miles. Para esto simplemente se debe finalizar el formato con la letra *c*. Además de los formatos presentados, hay otros cinco formatos numéricos que utilizan notación hexadecimal y binarios, los cuales no se presentan debido a su poco uso.

[8] Este formato lo veremos con especial atención durante la clase de fechas.

Todas las variables de texto tienen el formato *%#s,* donde # corresponde al número de caracteres que se presentan. En el caso el número de caracteres del formato de una variable de texto sea menor al número de caracteres de su tipo, el editor de datos mostrará unos puntos suspensivos en la variable indicando que hay más información de la que se muestra.

Stata justifica la información a la derecha automáticamente. Si se quiere justificar a la izquierda es necesario poner el signo menos (-) después del símbolo porcentaje (%). Esto aplica tanto para los formatos numéricos como los formatos de texto.

Finalmente, para cambiar una variable de formato se puede emplear el comando format. Esto normalmente se hace cuando hay variables que están en un formato que dificulta su visualización. Para ilustrar mejor esto se desarrollará un ejemplo.

Ej. 18: En la base de datos de la clase la variable *ingresos_hogar_jefe* tiene información del ingreso mensual del jefe del hogar en miles de pesos. Veamos esta variable en detalle.

Ej. 18 ⟶ br *ingresos_hogar_jefe*

Se puede ver que la información está en formato exponencial, lo cual dificulta su interpretación. Para facilitar la interpretación de la información de esta variable, lo se va a cambiar el formato de esta al formato general sin decimales, justificado a la izquierda, y utilizando la coma (,) como separador de miles. El comando para realizar esto es:

Ej. 19 ⟶ format %-12.0gc *ingresos_hogar_jefe*

Ahora, al emplear el comando browse de nuevo se puede ver el cambio en la forma de presentación de esta variable.

6. Conversión entre tipos de variables

Para convertir una variable numérica a un formato string se utiliza el comando tostring. Por ejemplo, la variable *muestreo* es un indicador para cada barrio en la que se recogió la información. Dado que es una variable de identificación es más útil utilizar un formato *string* en este caso. Al cambiar una variable numérica a un formato *string* es necesario especificar si queremos reemplazar la variable o si nuestra intención es mantener la original y crear otra que contenga esta misma información en tipo *string*. Para crear otra variable, se especifica la opción generar nueva variable [, gen (nombre de la nueva variable)]:

Ej. 20 ⟶ tostring muestreo, gen(muestreo2)

Para reemplazar, se especifica la opción [, replace]:

Ej. 21 ⟶ tostring *muestreo*, replace

También es posible cambiar el formato de una variable *string* a numérica. En la base de datos Base2.dta la variable *malt* toma el valor de 1 si el niño ha sido víctima de maltrato y 0 en caso contrario. Si deseamos saber el porcentaje de niños que son víctimas de algún tipo de maltrato, deberíamos correr el comando:

Ej. 22 ⟶ sum *malt*

Dado que la variable MALT tiene formato string, este comando no nos brinda la información que necesitamos. Para esto utilizamos el comando destring.

Ej. 23 ⟶ destring *malt*, gen(*malt2*)

Si la variable en formato *string* incluye algún carácter no-numérico el comando destring mostrará un error. Si el problema se debe a un carácter específico que aparece de forma sistemática, la opción [,ignore(char)] donde *char* es el carácter en cuestión resuelve el problema. En caso de que se desee ignorar todos los caracteres no numéricos, la opción [, force] resuelve el problema. No obstante, aquellas observaciones donde *Stata* encuentre alguno de estos caracteres será reemplazada en la nueva variable por un *missing value*.

7. Operaciones con variables de caracteres

Una gran ventaja de las variables tipo string es que es posible tomar parte de la información que aparece en una variable y unificarla con el contenido de otras variables. En la base de datos para la clase de hoy hay tres variables que identifican al hogar. Estas son muestreo, vivienda y hogar. La información se recogió en tres formularios, muestreo nos indica el barrio donde se recogió la información. vivienda nos indica el número de la vivienda en el municipio donde se recogió la información. Finalmente, dado que puede haber varios hogares viviendo en una misma vivienda, cada hogar tiene un número de identificación diferente. Si necesitamos un identificador de hogares, tenemos que unir las tres variables. Dos niños viven en el mismo hogar solamente si

tienen el mismo número de muestreo, vivienda y hogar. Para esto utilizamos el comando egen con la concat:

Ej. 24 ⟶ egen *id_hogar*=concat(*muestreo vivienda hogar*)

Ej. 25 ⟶ gen *id_hogar=muestreo+vivienda+hogar*

Este comando genera una nueva variable que es el resultado de concatenar las variables muestreo vivienda hogar. concat es una función que aplica a variables de tipo string. Hay muchas otras funciones de este tipo, algunas de las cuales estudiaremos más adelante.

La función substr hace lo contrario, nos permite tomar una parte de la información almacenada en una variable string. Podemos ver que la variable depmuni indica el departamento y el municipio en el que vive el niño. Los primeros dos dígitos nos indican el departamento en el que vive el niño y los siguientes nos indican el municipio. Si queremos obtener información únicamente del departamento en el que vive el niño, la función substr es útil. El comando tiene la siguiente estructura: gen newvariable=substr(X,n1,n2)

Este comando hace que *Stata* genere una variable con el nombre *newvariable* que contiene la información de la variable X desde la posición n1 hasta la posición n1+n2. En nuestro caso, debemos digitar el comando:

Ej. 26 ⟶ gen departamento=substr(*depmuni*,1,2)

De esta forma, la variable departamento tomará los primeros dos dígitos de la variable *depmuni*.