

2015

Curso Básico STATA

Javiera Vásquez Núñez
Investigadora Centro UC de Encuestas y
Estudios Longitudinales
19/08/2015



Contenido

Introducción	3
Capítulo I: Una pequeña introducción al programa	5
¿Cómo se ve STATA?	5
Capítulo II: Como organizar el trabajo en STATA	8
Capítulo III: ¿Cómo cargar una base de datos?	11
Capítulo IV: Leyendo los datos	18
Describe.....	19
Codebook	21
Edit	23
list; inspect; duplicates; count; order	24
Summarize	28
Tabulate.....	31
Capítulo V: Modificación de una base de datos.	38
Modificación de Variables de una base de datos	38
Unir bases de datos: merge y append	43
Condensar una base de datos: collapse	52
Cambiar la estructura de la base de datos: reshape.....	53
Eliminar variables: keep y drop	56
Crear variables: generate y egen	58
Capítulo VI: Aplicando lo aprendido I	62
Un ejemplo aplicado	62
Archivos log	68
Archivos do: para trabajar en forma programada en STATA.....	70
Capítulo VII: Aplicando lo aprendido II	78
Porcentaje de meses cotizados, trabajados y cesante	78
Capítulo VIII: Crear matrices para guardar los datos	86
Capítulo IX: Ciclos recursivos	89
Capítulo X: Análisis descriptivo de los datos	90
Distribución de frecuencias	90
Estadísticas descriptivas	92
Medidas de tendencia central	95

Medidas de dispersión	99
Medidas de desigualdad.....	101
Test de Hipótesis sobre la media poblacional	104
Test de diferencia de medias.....	108
Diferencia de medias de dos variables:	108
Diferencia de media entre grupos.....	110
Covarianza y Correlación	110
Factores de expansión (ponderadores)	115
Capítulo XI: Gráficos.....	118
Capítulo XII: Ayuda: Help.....	144

Introducción

Este documento pretende los introducir el uso del programa STATA, mediante la entrega de conceptos y comandos básicos para el inicio en este software. En una primera parte haremos una pequeña introducción al programa, luego veremos brevemente como se carga una base de datos, y en una tercera parte se verá cual es la manera más adecuada de trabajar con ella. Adicionalmente, en una cuarta parte se verán los comandos que comúnmente se utilizan para inspeccionar una base de datos y obtener estadísticas descriptivas de sus variables, específicamente veremos comandos como sum, el cual nos entrega un set de estadísticas básicas de la variable, describe, y varios otros. En una quinta parte se enseñara como se pueden modificar las bases de datos, especialmente, cambiando el nombre de las variables, unir bases de datos, eliminar y agregar variables, cambiar la estructura de ellas y crear nuevas variables a partir de las variables existentes.

En una sexta parte se aplicará lo aprendido y a partir de ese ejercicio se introducirá la utilidad de los archivos log y do. En la séptima parte se mostrará cómo podemos plasmar la información que nos interesa de los datos en diferentes tipos de gráficos. Finalmente en la octava y última parte se enseñará la utilidad del comando help, el cual nos será de gran utilidad para poder avanzar y utilizar stata cuando no tenemos muy claro cual es el comando a utilizar.

Es importante aclarar que gran parte de los comandos de stata se puede hacer mediante la utilización directa del comando, o mediante la utilización de ventanas. Durante este curso veremos ambas formas.

Un concepto importante de entender antes de comenzar a utilizar este software estadístico, es el de **Base de Datos**, el primer paso para poder trabajar con STATA es cargar la base de datos. Una base de datos en un conjunto (matriz) de información, tenemos filas y columnas, las que en su conjunto forman la base de datos. Generalmente se organiza de forma tal que las variables se representan por columnas y las observaciones por filas. Por ejemplo, si estamos estudiando las variables escolaridad e ingreso para las mujeres. Nuestra base de datos tendrá dos columnas, donde cada una de ellas representa la escolaridad e ingreso, y cada fila representa una mujer.

escolaridad	Ingreso
10	80.000
12	120.000
13	110.000
4	85.000
5	70.000
8	65.000
17	450.000
21	1.200.000
2	60.000

Capítulo I: Una pequeña introducción al programa

STATA es una aplicación completa e integrada, basada en comandos, que tiene todos los elementos necesarios para realizar análisis estadístico, manejo de datos estadísticos y gráficos. Las versiones mas nuevas de STATA (a partir de la versión 8.0) posee una forma más fácil de utilizar, que consiste simplemente en hacer clic en ventanas con las opciones de análisis y procesamiento de datos, además tiene la opción "antigua" mediante los comandos. El programa posee una ayuda en línea, es un programa fácil y rápido de utilizar.

¿Cómo se ve STATA?

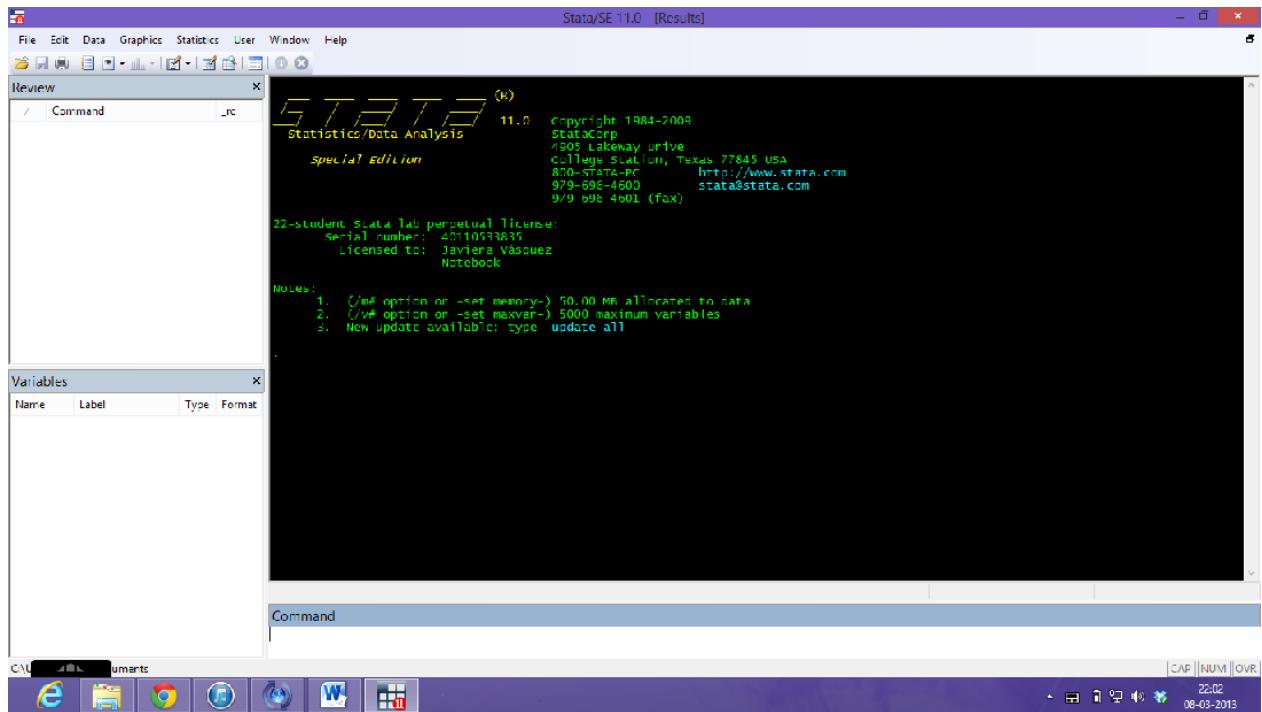
Cuando abrimos el programa, inmediatamente podemos distinguir 4 ventanas:

Review: en esta ventana aparecen los comandos que han sido utilizados durante la sección en turno.

Results: muestra los resultados de la aplicación de los comandos, sólo los resultados más recientes son visibles en esta ventana

Variables: en esta venta se presenta el listado de variables que se encuentran en la base de datos que se este trabajando

Commands: corresponde a la ventana donde introducen los comandos para obtener el resultado deseado. Sirve para utilizar STATA en forma interactiva.



Los íconos de la parte superior tienen los siguientes usos:

-  Abrir una base de datos
-  Guardar una base de datos, una vez que ha sido modificada en el programa
-  Imprimir los resultados de la ventana de resultados (STATA Results)
-  Comenzar o abrir un archivo **log**. Estos archivos tienen un formato de texto y permiten ir guardando todos los resultados.
-  Abrir el editor de do-file. Los archivos **do** son archivos con esta extensión que nos permiten en forma ordenada escribir todo lo que queremos hacer en nuestra base de datos: cambiar la base de datos, sacar estadísticas, etc..., y luego presionando  correr dicho do y obtener los resultados.¹

¹ Lo ideal es combinar la utilización de un **do** y un **log**; el primero permite tener en forma ordenada todos los comandos que se están utilizando y todas las instrucciones que se quieren ejecutar,

-  Permite ver y editar la Base de Datos.
-  Es igual al EDITOR, pero no permite eliminar variables ni observaciones.
-  Es el administrador de variables, aquí se puede nombrar y etiquetar las variables y sus categorías.
-  Es para detener la ejecución de un comando.

mientras que el segundo guarda en un archivo de texto todos los resultados que surgen de este archivo do.

Capítulo II: Como organizar el trabajo en STATA

Cuando se abre STATA es importante saber donde se esta trabajando, es decir, en que carpeta se están guardando los resultados o desde que carpeta vamos a llamar la base de datos, etc. Si no se sabe la carpeta o directorio donde STATA esta ubicado podemos averiguarlo escribiendo el comando **pwd**:

```
pwd
```

C:\data → Este resultado nos indica que estamos ubicados en el disco C del computador en la carpeta data

Para cambiar el directorio o carpeta se debe realizar lo siguiente:

```
cd C:\Nivelacion_Stata
```

Utilizo el comando **cd** y entrego la nueva ruta. En este caso le estoy indicando al programa que se ubique en la carpeta “Nivelacion_Stata” que se encuentra en el disco C del computador.

La ventaja de indicar desde un comienzo en que carpeta del computador se esta trabajando, es que evita indicar la ruta completa de los archivos cada vez que queramos abrir o guardar una base de datos, o abrir o guardar un log. Obviamente esto tiene sentido cuando para un trabajo específico tenemos todos los archivos necesarios en la misma carpeta.

Por ejemplo, si estamos trabajando con información de tres bases de datos distintas, y queremos dejar la información relevante para el estudio en una sola base datos (más adelante veremos como hacer esto), lo ideal es trabajar

en una sola carpeta, "Nivelacion_Stata", y no tener las tres bases de datos repartidas en carpetas distintas. Si no están en la misma carpeta no es útil indicarle el directorio al comienzo, ya que igual cuando llamemos a cada una de las bases de datos, al estar en carpetas distintas, tendremos que cambiar la ruta.

Importante: los sistemas operativos más nuevos permiten que las carpetas tengan nombres con espacio en blanco, por ejemplo, "Nivelacion Stata". Sin embargo, STATA no va a reconocer una carpeta que tenga **espacios en blanco** en el nombre, a no ser que se indique la ubicación de esta carpeta entre comillas. Por este motivo, se debe **evitar** llamar a una carpeta con la que van a trabajar en STATA con nombres que contengan espacios en blanco.²

Supongamos que la carpeta en que vamos a tratar se llama "Nivelacion Stata", en la primera línea del siguiente cuadro podemos apreciar que al entregar la ubicación de la carpeta utilizando el comando `cd`, el programa nos entrega un error "invalid syntax", esto se debe a que el nombre de la carpeta tiene espacios en blanco. Si agregamos comillas a la ruta no se produce el error.

```
. cd C:\Javiera\cursos_magister\MPP\Nivelacion de Stata\2013  
invalid syntax  
r(198);  
  
. cd "C:\Javiera\cursos_magister\MPP\Nivelacion de Stata\2013"  
C:\Javiera\cursos_magister\MPP\Nivelacion de Stata\2013  
  
. pwd  
C:\Javiera\cursos_magister\MPP\Nivelacion de Stata\2013
```

Si al introducir un comando **no** aparece un punto blanco después de ejecutado el comando, significa que no se terminó o no se ha terminado de ejecutar. Además, siempre que aparezcan letras rojas significa que hay un error, la ayuda para el error la pueden encontrar pinchando `r(198)`.

² Esto problema es común cuando trabajan en el Escritorio del computador, ya que la carpeta en este caso es C:\Documents and Settings\...., tiene espacios en blanco.

En resumen, para trabajar ordenadamente en STATA es conveniente crear una carpeta para cada trabajo independiente, esta carpeta debe tener una ruta que no contenga espacios en blanco en los nombres.

Capítulo III: ¿Cómo cargar una base de datos?

Las bases de datos en formato Stata tienen extensión .dta. Las versiones antiguas del software no se pueden abrir bases de datos que han sido trabajadas y guardadas en una versión más moderna, cuando intentemos hacer esto el programa entregará un error indicando que la base no tiene formato Stata.

Antes de abrir una base de datos se tienen que cumplir dos condiciones:

1. El programa debe estar limpio, sin ninguna base de datos ya cargada. Para limpiar el programa de otras bases de datos se debe utilizar el comando **clear**. Si he estado trabajando una base de datos previamente la cual se ha modificado y no he guardado estas modificaciones, al intentar abrir una nueva base de datos sin limpiar antes arrojará el siguiente error:

```
no; data in memory would be lost
```

2. El programa debe tener suficiente memoria. Para entregarle memoria a Stata se debe utilizar el comando **set mem**. Por ejemplo, si la base de datos que deseamos cargar pesa 100 MB, en la ventana **Stata Command** debemos tipar:

```
set mem 100m
```

Si Ud. no agrega memoria y los 10 MB que vienen asignados al abrir el programa no son suficientes, el programa arrojará el siguiente error:

no room to add more observations

Esto también puede suceder cuando se ha trabajado en la base de datos y se han creado muchas variables: en un momento el programa se puede quedar sin memoria. En este caso se debe limpiar el programa (borrar la base de datos) utilizando el comando **clear**; entregarle más memoria al programa utilizando **set mem**; abrir la base de datos y realizar todo nuevamente. Por esta razón es fundamental que Ud., cuando comience a trabajar, asigne la memoria necesaria para todas las variables que espera generar.

El comando general para entregar memoria a Stata es:

set mem # [b|k|m|g] [, permanently]

con la opción “permanently” la cantidad de memoria ingresada se mantendrá cada vez que se inicie nuevamente el programa.

Existen distintas formas de cargar una base de datos:

- 1- Utilizando una base ya grabada con la extensión de STATA, es decir, disponer de la base de datos como **nombre.dta** En este caso podemos apretar el icono  y buscar la ubicación de la base de datos. También podemos hacerlo dirigiéndonos a File/Open...
- 2- Otra forma es tipar en **Stata Command** use “[disco en que la guardaremos] \ [ruta de acceso] \ [nombre de archivo.dta]”, **clear**. Por ejemplo:

```
use "C:\Users\Javiera\Dropbox\Cursos de Capacitación\Curso Básico Stata Desarrollo Social\Bases\exteps09.dta", clear
```

o simplemente

```
use "Bases\exteps09.dta", clear
```

si ya le hemos indicado previamente a Stata que vamos a trabajar en la carpeta “Curso Básico Stata Desarrollo Social” ubicada en del disco C:\Users\Javiera\Dropbox\Cursos de Capacitación. Lo que previamente determinamos con el siguiente comando:

```
cd "C:\Users\Javiera\Dropbox\Cursos de Capacitación\Curso Básico Stata Desarrollo Social"
```

Notar que en ambos casos el comando incorpora la opción “, clear”, esto nos garantiza que la base de datos sea abra si es que ya existe otra base de datos previa en el programa, esta opción ahorra el paso previo de ejecutar el comando **clear** antes de abrir la base de datos.

Recuerde que si la carpeta en la que está trabajando tiene espacios en blanco, debo poner comillas al llamar la base de datos, de lo contrario aparecerá el siguiente error:

```
. use C:\Users\Javiera\Dropbox\Cursos de Capacitación\Curso Básico Stata Desarrollo Social\Bas  
> es\exteps09.dta, clear  
invalid 'de'  
r(198);
```

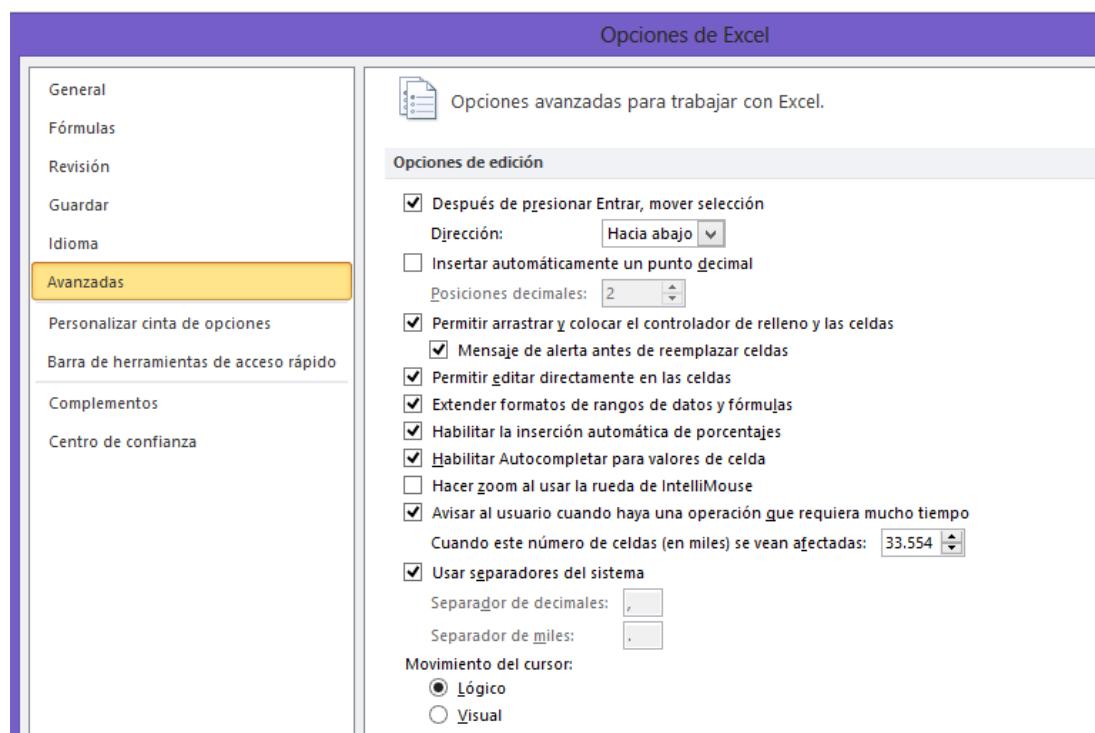
Esto porque Stata cree que el nombre de la carpeta es simplemente Cursos. Si utilizamos comillas no se produce el error.

3- Traspasar los datos de un archivo Excel o similar copiando la información de este archivo al EDITOR de STATA.

Esto se hace copiando en el archivo Excel las columnas (variables) que queremos traspasar como base de datos a STATA (Ctr+C). Luego nos dirigimos a STATA abrimos el EDITOR  y pegamos la información (Ctr+V). Obviamente antes de hacer esto se debe haber limpiado Stata con el comando **clear**.

Algunos aspectos relevantes antes de copiar los datos de Excel a Stata:

Para Stata, como para cualquier otro software norteamericano, el separador de miles es la coma (,), y el separador de decimales es el punto (.); Si el computador en el que esta trabajando no esta configurado de esta forma, debe dirigirse en Excel a “Archivo” luego pinchar “Opciones”:



En “Avanzadas” se puede modificar los separadores del sistema, hay que utilizar “.” para decimales y “,” para el separador de miles.

Todas las variables que son numéricas, deben estar en formato numérico antes de ser exportadas.

El siguiente cuadro muestra lo que resulta de pasar la base de datos base.xls a Stata:

	ao	mes	afp	cotizacina~l	comisinfij~n	comisinfijo~o	comisinpor~e	
1	1981	5	Alameda	.026	228	.	.006	
2	1981	5	Concordia	.026	3	.	.025	
3	1981	5	Cuprum	.026	.	.	.025	
4	1981	5	El Libertador	.028	.	16	.015	
5	1981	5	Habitat	.025	.	12	.012	
6	1981	5	Invierta	.024	12	.	.017	
7	1981	5	Magister	
8	1981	5	Planvital	.025	139	.	.019	
9	1981	5	Provida	.026	.	15	.008	
10	1981	5	San Cristóbal	.025	98	.	.019	
11	1981	5	Santa María	.025	.	.	.021	
12	1981	5	Summa	.027	.	96	.017	
13	1981	6	Alameda	.026	228	.	.006	
14	1981	6	Concordia	.026	3	.	.025	
15	1981	6	Cuprum	.026	.	.	.025	
16	1981	6	El Libertador	.028	.	16	.015	
17	1981	6	Habitat	.025	.	12	.012	
18	1981	6	Invierta	.024	12	.	.017	
19	1981	6	Magister	

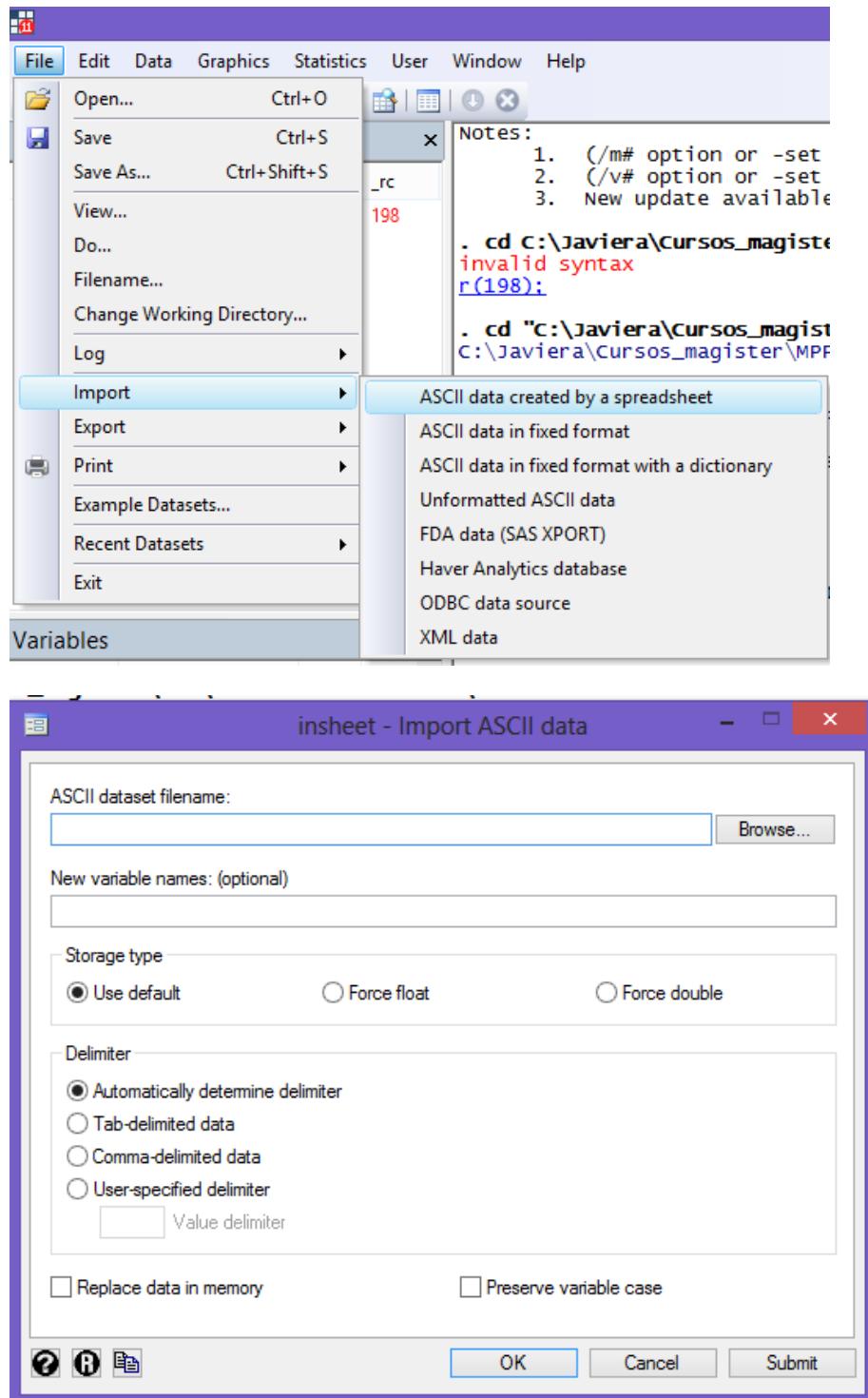
El color rojo indica que la variable no es numérica.

Generalmente las bases de datos muy grandes no vienen en Excel, ya que este programa es limitado en cuanto al número de filas (observaciones) y número de columnas (variables). El número máximo de filas es de 65.536, y el número máximo de columnas es de 256.

- 4- Otra forma de cargar bases de datos es mediante el comando **insheet**, este comando permite cargar bases de datos en formato ASCII (texto) mediante el siguiente comando:

```
. insheet using Bases\junio05.txt  
(102 vars, 11158 obs)
```

o alternativamente:



Cuando las bases de datos vienen en texto y son muy grandes no se pueden ver utilizando un block de notas, en estos casos se recomienda utilizar el programa TextPad que puede ser descargado gratuitamente

(www.textpad.com). Siempre es recomendable inspeccionar la base de datos en texto antes de ser traspasada a Stata.

- 5- Si la base de datos tiene otro formato, por ejemplo, SPSS (.sav), dbase (.dbf), Access (.mbd), etc; existe un software llamado **Stat Transfer**, que permite transformar base de datos desde y a diversos formatos.

Luego para guardar la base de datos utilizamos el comando **save**:

- 1- Si quiere reescribir la base de datos antigua:

```
. save "Bases\exteps09.dta", replace  
file Bases\exteps09.dta saved
```

Es importante escribir **replace**, sino el programa les enviara un error diciendo que la base de datos ya existe.

- 2- Si quiere guardar la base de datos con un nuevo nombre no es necesario tipear replace:

```
. save "Bases\exteps09_2.dta", replace  
(note: file Bases\exteps09_2.dta not found)  
file Bases\exteps09_2.dta saved
```

Una vez que los datos han sido cargados, se puede optimizar el espacio que estos ocupan utilizando el comando **compress**, este comando comprime la base de datos. Es muy útil cuando trabajamos con bases de datos grandes.

Hasta ahora hemos aprendido como cargar una base de datos en Stata, en lo que sigue se verán los comandos básicos para analizar una base de datos.

Entonces, con los comandos recién estudiados, comenzemos por abrir la base de datos:

```
. clear  
. set mem 100m
```

Current memory allocation

settable	current value	description	memory usage (1M = 1024k)
set maxvar	5000	max. variables allowed	1.947M
set memory	100M	max. data space	100.000M
set matsize	400	max. RHS vars in models	1.254M
			103.201M

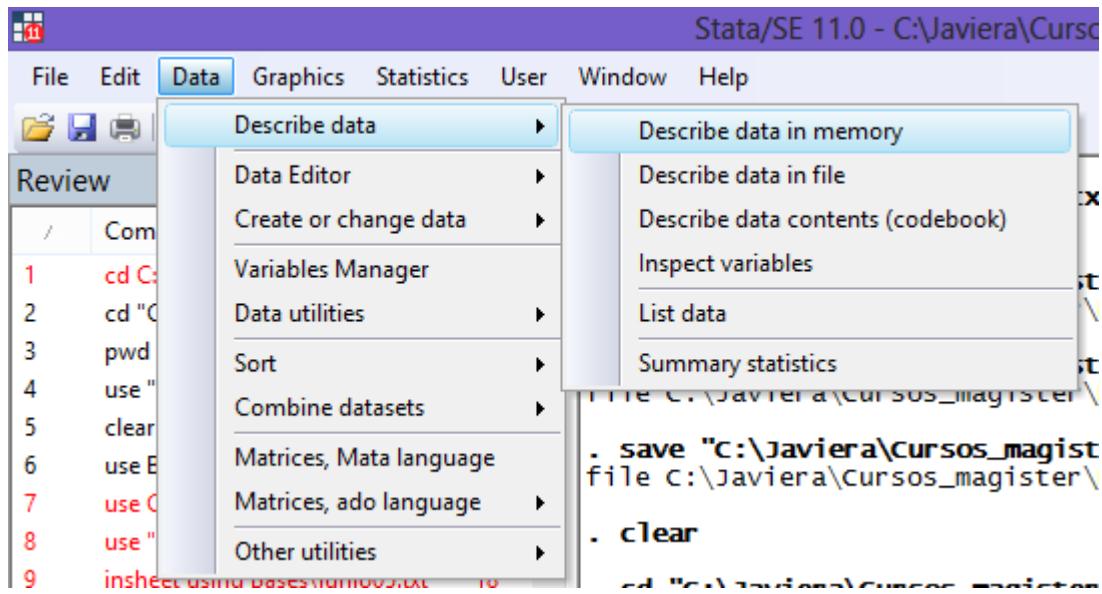
```
. cd "C:\Javiera\Cursos_magister\MPP\Nivelacion de Stata\2013"  
C:\Javiera\Cursos_magister\MPP\Nivelacion de Stata\2013  
. use Bases\exteps09.dta
```

Capítulo IV: Leyendo los datos

Existen varios comandos que nos permiten obtener información preliminar acerca de los datos y estadísticas acerca de ellos. Veremos básicamente los comandos `describe`, `codebook`, `edit`, `sum`, `tab`, `inspect`, `count` y `duplicates`.

Antes de ver detalladamente cada uno de estos comandos descriptivos es necesario aclarar que cada uno de ellos puede ser utilizado para ver el comportamiento de grupos de observaciones mediante las opciones `by` e `if`. Por ejemplo: `by sexo: codebook`, el cual nos mostrara la aplicación del comando `codebook` separado para cada sexo o `codebook if sexo==1`, el cual nos mostrará la aplicación del comando `codebook` para el sexo que este codificado con el número 1.

También podemos leer los datos usando las ventanas correspondientes, tal como lo observamos en la siguiente figura:



Como podemos ver, si vamos a la opción **Data** y luego seguimos la opción **Describe data**, veremos una serie de opciones que veremos a continuación, tal como el comando describe o el codebook. También mediante la opción Data podemos directamente ver el editor de stata.

Describe

El comando **describe** entrega información de todas las variables que se encuentran en la base de datos. Esta información incluye el tipo de almacenamiento (byte, int, long, float, double, string)³, el formato de los datos, la variable que contiene el label (etiqueta), y la descripción de la variable. Además entrega información de número de observaciones, número de variables y tamaño de la base de datos.

³ Ver Anexo A sobre el tipo de almacenamiento de datos

. describe

```
Contains data from Bases\exteps09.dta
obs:      14,243
vars:      29
size:  1,922,805 (98.2% of memory free)
```

variable name	storage type	display format	value label	variable label
folio	double	%9.0g		
sexo09	byte	%9.0g		a8. sexo
edad09	int	%9.0g		a9. ¿qué edad tiene ud.?
b2_09	float	%32.0g	b2	Estatus Laboral Abril 2009
b4_09	float	%32.0g	b4	Region en que trabaja (Abril 2009)
b6_09	float	%32.0g	b6	Tipo de trabajo (Abril 2009)
b8_09	float	%32.0g	b8	Categoría Ocupacional (Abril 2009)
b9a_09	float	%9.0g	b9a	Tiene contrato (Abril 2009)
b9b_09	float	%32.0g	b9b	Relación Contractual (Abril 2009)
b10_09	float	%40.0g	b10	Horario (Abril 2009)

Cuando la base de datos es muy grande y sólo se quiere obtener información de algunas de las variables contenidas en ella, después de **describe** (o simplemente **d**) se ingresa la lista de variables de las cuales Ud. desea una descripción.

Otras formas de utilizar el comando **describe**:

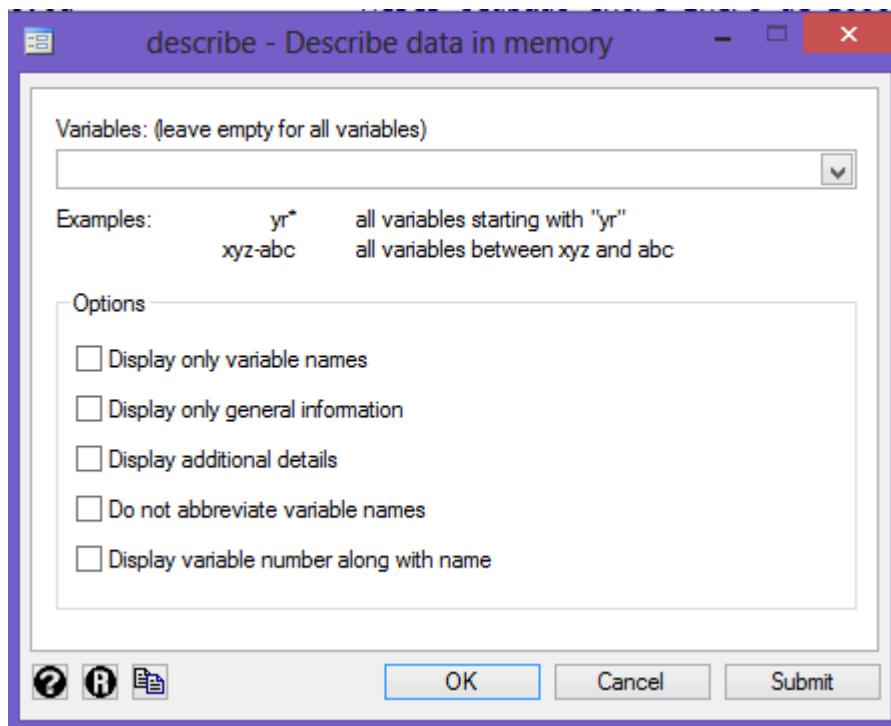
. describe e*

variable name	storage type	display format	value label	variable label
edad09	int	%9.0g		a9. ¿qué edad tiene ud.?
esc09	float	%9.0g		Años de escolaridad EPS09

. describe ocu_0609- pcot_ocu

variable name	storage type	display format	value label	variable label
ocu_0609	double	%9.0g		Meses ocupado entre Enero de 2006 y Abril de 2009
ces_0609	double	%9.0g		Meses cesante entre Enero de 2002 y Septiembre de 2006
inact_0609	double	%9.0g		Meses inactivo entre Enero de 2006 y Abril de 2009
cot_0609	double	%9.0g		Meses cotizados entre Enero de 2006 y Abril de 2009
pcot	float	%9.0g		Proporción del tiempo cotizado entre Enero 2004 y Septiembre 2004
pcot_ocu	float	%9.0g		Proporción del tiempo ocupado cotizado entre Enero 2004 y Septiembre 2004

Para usar el comando mediante el uso de ventanas basta con seguir el mismo esquema recién mostrado y hacer doble clic sobre **describe variables in memory**.



Luego se escribe el nombre de la variable de la cual quiero la descripción y si quisiese, por ejemplo, una descripción específica, puedo aplicar alguna de las opciones que se observan en la ventana anterior.

Codebook

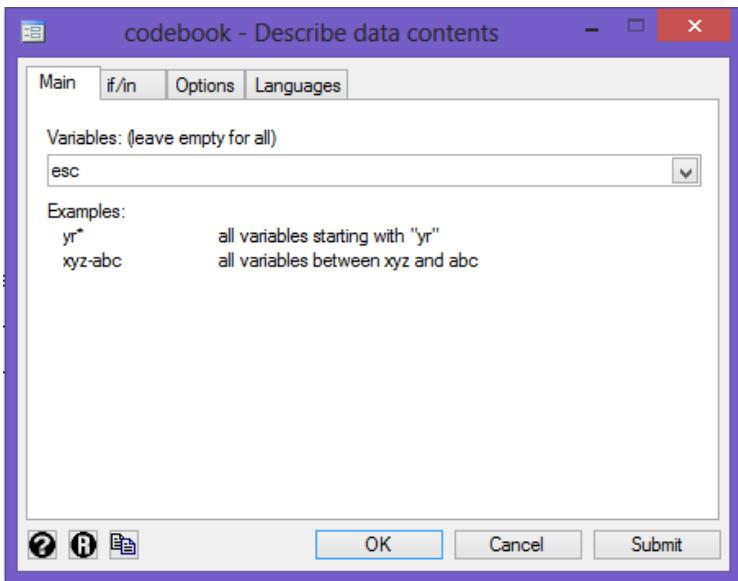
Existen otras formas de obtener una descripción de las variables en la base de datos, una de las mejores es usar el comando `codebook`. Al igual que con el comando `describe`, al tipar simplemente `codebook` se va a describir cada variable. El siguiente cuadro muestra la diferencia entre ambos comandos:

. codebook esc	
esc09	Años de escolaridad EPS09
type: numeric (float)	
range: [0,21]	units: 1
unique values: 22	missing .: 119/14243
mean: 9.69662	
std. dev: 4.32418	
percentiles:	10% 25% 50% 75% 90%
	3 6 12 12 15

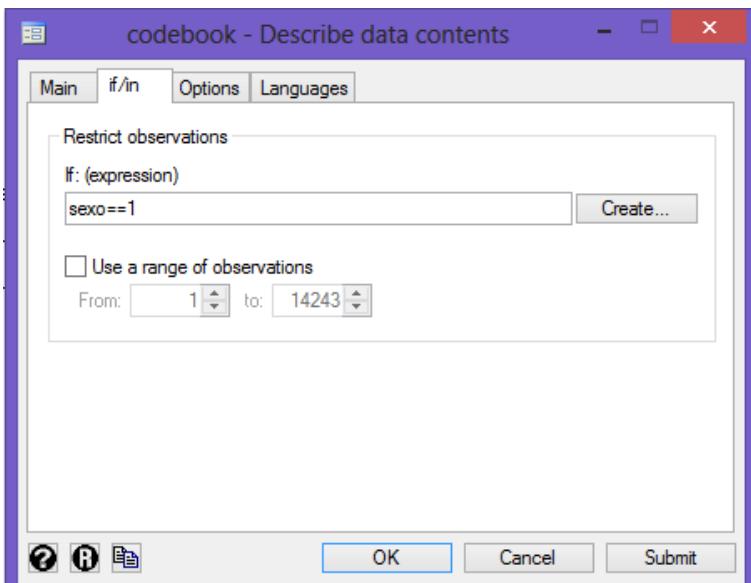
Claramente el comando **codebook** es más completo, presenta la etiqueta de la variable, el formato, el rango de los datos, si esta codificada en números enteros (units: 1), cuantas observaciones no tienen dato de esta variable, el promedio, la desviación estándar, y los percentiles.

Tanto el comando **describe** como el comando **codebook** nos permite distinguir dos tipos de variables, las numéricas y las no numéricas. Sólo se pueden obtener estadísticas de los datos cuando las variables son numéricas, aunque muchas veces es más fácil visualizar la base de datos cuando las variables tienen nombres en vez de números o códigos.

También podemos aplicar el comando vía la utilización de ventanas, lo que es similar al caso de describe, con la salvedad de que el doble click se hace sobre "Describe data contents (codebook)". Ahí se pone el nombre de la variable que queremos inspeccionar:



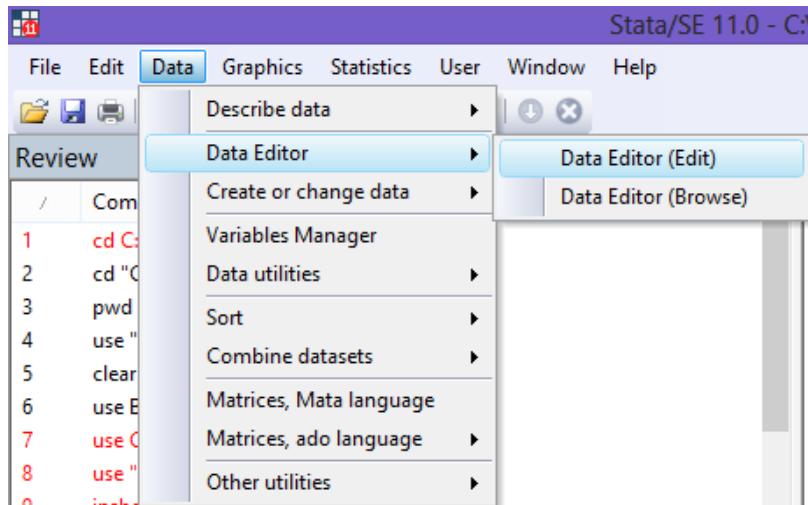
Otra función que podemos aplicar sobre la misma ventana es condicionarla a algo, como, por ejemplo, que la inspección sea sólo para lo hombres:



Edit

Otra forma de conocer o tener una visión más amplia de la base de datos es mediante el comando `edit`; el que nos mostrará una planilla donde podremos ver la base de datos completa; sus variables y todas sus observaciones.

También lo podemos hacer vía ventanas mediante la opción “Data” y luego “Data editor”



Al igual que en los casos anteriores también se puede realizar la misma operación vía el comando edit y también podemos editar una matriz mas pequeña especificando las variables que queramos incluir utilizando la encuesta casen 2003, por ejemplo “edit sexo09 edad09 esc09 numer ytotal”

The screenshot shows the Stata Data Editor window titled 'Data Editor (Edit) - [exteps09]'. The window has a menu bar with 'File', 'Edit', 'Data', and 'Tools'. Below the menu is a toolbar with various icons. The main area displays a dataset named 'var32[21]'. The table has five columns: 'sexo09', 'edad09', 'esc09', 'numer', and 'ytotal'. The data consists of 12 rows, each containing values for these variables. The 'numer' column contains values like 5, 4, 5, 4, etc., and the 'ytotal' column contains large numerical values such as 3072000, 696000, 2538360, etc.

	sexo09	edad09	esc09	numer	ytotal
1	1	72	6	5	3072000
2	1	48	12	4	696000
3	1	44	10	5	2538360
4	1	26	8	4	2940000
5	1	44	17	4	3.72e+07
6	1	53	12	5	8832000
7	1	35	12	5	2106000
8	1	65	3	5	696000
9	1	40	14	4	1.09e+07
10	1	25	15	2	0
11	1	24	15	5	2.23e+07
12	1	51	8	5	5592800

list; inspect; duplicates; count; order

Un comando bastante útil, similar al `edit`; es el `list`; el cual despliega los datos en la ventana de resultados (Stata Results); en vez de enviarnos al editor.

```
. list sexo09 edad09 esc09
```

	sexo09	edad09	esc09
1.	1	72	6
2.	1	48	12
3.	1	44	10
4.	1	26	8
5.	1	44	17
6.	1	53	12
7.	1	35	12
8.	1	65	3
9.	1	40	14
10.	1	25	15
11.	1	24	15
12.	1	51	8
13.	1	31	12
14.	1	32	14
15.	1	39	12
16.	1	47	15
17.	1	63	5
18.	1	60	.
19.	1	74	6
20.	1	27	12

Otro comando; el cual asegura que una variable es una codificación única dentro de una base de datos, es el comando `duplicates report`. Este comando se usa generalmente para chequear que no existan observaciones duplicadas (folios duplicados) dentro de una base de datos. Si tenemos la siguiente base de datos:

```
. clear  
. use Bases/ejdupli.dta
```

	id	esc
1	1	8
2	1	8
3	1	8
4	2	9
5	3	12
6	3	12
7	4	15
8	5	4
9	6	8
10	7	11
11	7	11
12	7	11
13	7	11
14	7	11
15	7	11
16	8	10
17	8	10
18	9	18

El resultado que entrega aplicar este comando a la variable “id” es el siguiente:

```
. duplicates report id
duplicates in terms of id
```

copies	observations	surplus
1	5	0
2	4	2
3	3	2
6	6	5

Finalmente, existen otros dos comandos interesantes para inspeccionar la base de datos: **inspect** y **count**.

El comando **inspect** muestra la distribución de la variable, la cantidad de observaciones con valor cero, con valores mayores a cero y sin dato, así como la cantidad de número enteros y no enteros en la variable. Por ejemplo:

```
. use Bases\exteps09.dta, clear
. inspect esc09
```

esc09: Años de escolaridad EPS09

		Number of Observations		
		Total	Integers	Nonintegers
#	Negative	-	-	-
#	Zero	413	413	-
#	Positive	13711	13711	-
# #	Total	14124	14124	-
# # #	Missing	119		
0		14243		
(22 unique values)				
21				

El comando **count**, lo que hace es contar tal como lo dice su nombre. Por ejemplo:

```
. count
14243

. count if sexo09==1
6950

. count if ing_entrev_mensual==0
2392

. count if ing_entrev_mensual>0
11851

. count if esc09<=8
5855

. display 6950/14243
.487959

. di 2392/14243*100
16.794215
```

El comando **display** (**di**) funciona como calculadora.

Un comando adicional; que si bien no sirve para inspeccionar la base de datos, si no ayuda a inspeccionarla; ya que nos permite ordenar las variables de la manera que más nos acomode. Este comando es **order** el cual se utiliza de la siguiente manera:

```
. order folio edad09 esc09 sexo09
```

y nos entregará la base de datos de tal forma que al aplicar el comando edit la primera variable que veremos será folio, luego edad, luego años de escolaridad, etc.

Summarize

El comando **summarize**, el que se abrevia **sum**, entrega estadísticas básicas: número de observaciones, promedio, desviación estándar, mínimo y máximo, de las variables que se especifiquen. Si sólo se escribe **sum** en Stata Command, se muestran las estadísticas de todas las variables en la base de datos.

```
. sum
```

Variable	Obs	Mean	Std. Dev.	Min	Max
folio	14243	1393429	798800.4	179	2419215
edad09	14243	49.93239	15.22389	19	108
esc09	14124	9.696616	4.324177	0	21
sexo09	14243	1.512041	.4998725	1	2
b2_09	14241	2.069798	1.372703	1	4
b4_09	8309	9.345529	4.153978	1	99
b6_09	8309	1.345168	.8506744	1	5
b8_09	8309	3.407991	1.119213	1	8
b9a_09	6194	1.310462	.8203749	1	9
b9b_09	5361	1.365603	.9335594	1	9
b10_09	6194	1.335647	.8291077	1	9
b11_09	8281	1.964497	.6212386	1	9
b12_09	8285	286148.8	444763	0	1.50e+07
b13_09	8309	94.71645	211.105	2	999
b14_09	8309	5.358045	1.621125	1	9
b15_09	8309	25985.25	43463.92	0	99999
b15t_09	2143	42.08259	44.12646	1	99
b16_09	8309	1.925262	.8601393	1	9
b17_09	8309	3.60621	2.864865	1	9
b18_09	8309	2.753641	2.732953	1	9

Si nos interesan las estadísticas de una sola variable, por ejemplo, escolaridad:

```
. sum esc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
esc09	14124	9.696616	4.324177	0	21

Si requerimos las estadísticas de escolaridad, pero separado para hombres y mujeres, primero se debe ordenar la base de datos por género (`sort dhombre`) y luego hacer un `sum` utilizando el comando `by`:

```
. by sexo: sum esc  
not sorted  
r(5);
```

```
. bys sexo: sum esc
```

```
-> sexo09 = 1
```

variable	obs	Mean	Std. Dev.	Min	Max
esc09	6897	9.664492	4.307995	0	21

```
-> sexo09 = 2
```

variable	obs	Mean	Std. Dev.	Min	Max
esc09	7227	9.727273	4.33964	0	21

Esto mismo se podría hacer alternativamente utilizando `if` :

```
. sum esc09 if sexo==1
```

variable	obs	Mean	Std. Dev.	Min	Max
esc09	6897	9.664492	4.307995	0	21

```
. sum esc09 if sexo==2
```

variable	obs	Mean	Std. Dev.	Min	Max
esc09	7227	9.727273	4.33964	0	21

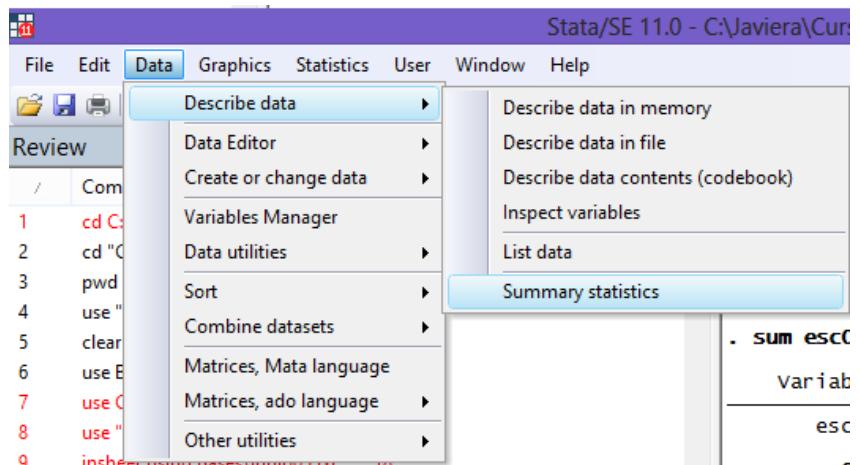
Si al comando `sum` le agregamos `detailed` después de una ","; STATA entrega una cantidad más amplia de estadísticas sobre la variables. Además de las ya descritas entrega los percentiles, la varianza, la asimetría y la kurtosis.

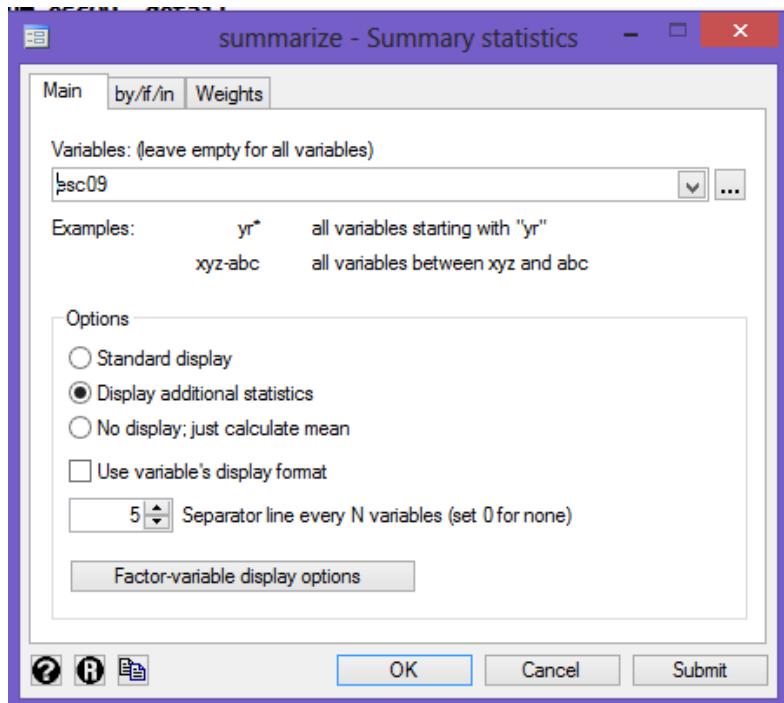
```
. sum esc09, detail
```

Años de escolaridad EPS09

	Percentiles	smallest		
1%	0	0		
5%	2	0		
10%	3	0	obs	14124
25%	6	0	sum of wgt.	14124
50%	12		Mean	9.696616
		Largest	Std. Dev.	4.324177
75%	12	21		
90%	15	21	variance	18.69851
95%	17	21	Skewness	-.3412399
99%	17	21	Kurtosis	2.465894

Todo lo anterior se puede hacer también mediante ventanas, especificaremos sólo un caso, donde se pide un sum de la variable esc y se pide una descripción detallada, lo que se obtiene mediante la opción “Display additional statistics”





Tabulate

El comando **tabulate** (o **tab**) permite hacer tablas con las variables de interés. La tabla más sencilla se realiza de la siguiente forma:

```
. tab sexo09
```

a8. sexo	Freq.	Percent	Cum.
1	6,950	48.80	48.80
2	7,293	51.20	100.00
Total	14,243	100.00	

Esta tabla indica el número de observaciones total y de cada una de las categorías de la variable, el porcentaje que cada uno representa sobre el total y el porcentaje acumulado.

El comando `tab1` permite hacer esto mismo pero para varias variables simultáneamente:

```
. tab1 sexo09 b2_09
-> tabulation of sexo09
```

a8. sexo	Freq.	Percent	Cum.
1	6,950	48.80	48.80
2	7,293	51.20	100.00
Total	14,243	100.00	

Estatus laboral Abril 2009	Freq.	Percent	Cum.
trabajando	8,309	58.35	58.35
cesante	1,274	8.95	67.29
buscando trabajo por 1ra. vez	13	0.09	67.38
inactivo	4,645	32.62	100.00
Total	14,241	100.00	

Además, con este comando, se pueden realizar cruces entre variables, por ejemplo:

```
. tab b2_09 sexo
```

Estatus laboral Abril 2009	a8. sexo		Total
	1	2	
trabajando	5,060	3,249	8,309
cesante	534	740	1,274
buscando trabajo por 1ra. vez	5	8	13
inactivo	1,350	3,295	4,645
Total	6,949	7,292	14,241

Si en vez de las frecuencias uno quiere ver el porcentaje, que sume 100% en forma horizontal (filas), se debe agregar a lo anterior una coma y la palabra `row` y poner además `nofreq` (para que no se muestre las frecuencias):

```
. tab b2_09 sexo, row nofreq
```

Estatus laboral Abril 2009	a8. sexo		Total
	1	2	
trabajando	60.90	39.10	100.00
	41.92	58.08	100.00
	38.46	61.54	100.00
	29.06	70.94	100.00
Total	48.80	51.20	100.00

Si se quiere que los porcentajes sumen 100% en forma vertical (columnas) debemos hacer lo siguiente:

```
. tab b2_09 sexo, col nofreq
```

Estatus laboral Abril 2009	a8. sexo		Total
	1	2	
trabajando	72.82	44.56	58.35
	7.68	10.15	8.95
	0.07	0.11	0.09
	19.43	45.19	32.62
Total	100.00	100.00	100.00

Si deseamos que se muestren ambos porcentajes:

```
. tab b2_09 sexo, col row nofreq
```

Key
row percentage
column percentage

Estatus laboral Abril 2009	a8. sexo		Total
	1	2	
trabajando	60.90	39.10	100.00
	72.82	44.56	58.35
cesante	41.92	58.08	100.00
	7.68	10.15	8.95
buscando trabajo por	38.46	61.54	100.00
	0.07	0.11	0.09
inactivo	29.06	70.94	100.00
	19.43	45.19	32.62
Total	48.80	51.20	100.00
	100.00	100.00	100.00

También se pueden hacer tablas utilizando el comando **by**, primero ordenando de acuerdo a la variable que voy a realizar las tablas, por ejemplo:

```
. bys sexo09: tab b2_09
```

```
-> sexo09 = 1
```

Estatus laboral Abril 2009	Freq.	Percent	Cum.
buscando trabajo por 1ra. vez inactivo	5,060	72.82	72.82
	534	7.68	80.50
	5	0.07	80.57
	1,350	19.43	100.00
Total	6,949	100.00	

```
-> sexo09 = 2
```

Estatus laboral Abril 2009	Freq.	Percent	Cum.
buscando trabajo por 1ra. vez inactivo	3,249	44.56	44.56
	740	10.15	54.70
	8	0.11	54.81
	3,295	45.19	100.00
Total	7,292	100.00	

Este código entrega dos tablas de la variable escolaridad, una para los hombres y otra para las mujeres. Lo mismo es posible de ser realizado utilizando el condicional **if**:

```
. tab b2_09 if sexo==1
```

Estatus laboral Abril 2009	Freq.	Percent	Cum.
buscando trabajo por 1ra. vez inactivo	5,060	72.82	72.82
	534	7.68	80.50
	5	0.07	80.57
	1,350	19.43	100.00
Total	6,949	100.00	

```
. tab b2_09 if sexo==2
```

Estatus laboral Abril 2009	Freq.	Percent	Cum.
buscando trabajo por 1ra. vez inactivo	3,249	44.56	44.56
	740	10.15	54.70
	8	0.11	54.81
	3,295	45.19	100.00
Total	7,292	100.00	

También se puede utilizar el comando `tabulate` para generar variables dicotómicas. Por ejemplo, si se quiere generar variables dicotómicas para cada situación laboral:

$$DEstatus_1 = \begin{cases} 1 & \text{si la persona trabaja} \\ 0 & \text{si no} \end{cases}$$

$$DEstatus_2 = \begin{cases} 1 & \text{si la persona esta cesante} \\ 0 & \text{si no} \end{cases}$$

⋮

Se puede hacer de la siguiente forma:

```
. tab b2_09, generate(DEstatus_)
```

Estatus laboral Abril 2009	Freq.	Percent	Cum.
trabajando	8,309	58.35	58.35
cesante	1,274	8.95	67.29
buscando trabajo por 1ra. vez	13	0.09	67.38
inactivo	4,645	32.62	100.00
Total	14,241	100.00	

Como esta variable tiene cuatro categorías diferentes, se crean cuatro variables binarias o dicotómicas.

Supóngase ahora que se requiere hacer un cuadro con los años de escolaridad, pero en vez de entregar la frecuencia o porcentaje de observaciones en cada categoría, se requiere que muestre una estadística de otra variable, por ejemplo, el promedio de ingreso. Para realizar este tipo de tablas se debe usar el comando `tab` agregándole después de una coma la palabra `summarize`. Para el ejemplo citado debería utilizar el siguiente código:

```
. tab esc09, summarize(ing_entrev_mensual) means
```

Años de escolaridad EPS09	Summary of ing_entrev_ mensual Mean
0	101186.33
1	75201.093
2	102586.96
3	115778.54
4	129247.77
5	114401.1
6	128952.18
7	167105.26
8	145126.82
9	178890.03
10	231746.11
11	247240.56
12	217041.55
13	371306.02
14	370284.47
15	370182.99
16	476301.6
17	692630.93
18	960761.91
19	941410.25
20	800855.75
21	1840060
Total	230243.43

Por último, el comando **tabstat** permite realizar tablas con las siguientes estadísticas:

mean	promedio
count	cuenta el número de observaciones que tiene valor igual que count
n	suma
sum	máximo
max	mínimo
min	rango=max-min
range	desviación estandar
sd	varianza
var	coeficiente de variacion (sd/mean)
cv	error estandar de la media = sd/sqrt(n)
semean	asimetria
skewness	kurtosis
kurtosis	mediana (lo mismo que p50)
median	1st percentile
p1	5th percentile
p5	10th percentile
p10	25th percentile
p25	50th percentile (igual que la mediana)
p50	75th percentile
p75	90th percentile
p90	95th percentile
p95	99th percentile
p99	rango interquartil = p75 - p25
iqr	equivalente a especificar "p25 p50 p75"
q	-----

Al poner `by(nombre)` especifica que las estadísticas deben ser entregadas separadamente para cada valor de la variable "nombre".

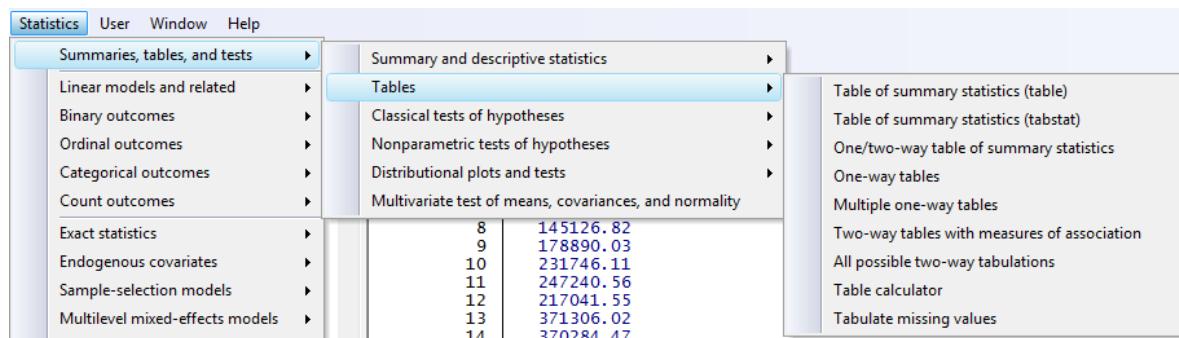
Por ejemplo, si se quiere obtener el promedio, la mediana, el máximo y el mínimo de ingreso por género, se puede realizar de la siguiente forma:

```
. tabstat ing_entrev_mensual, statistics(mean p50 min max) by(sexo09)
```

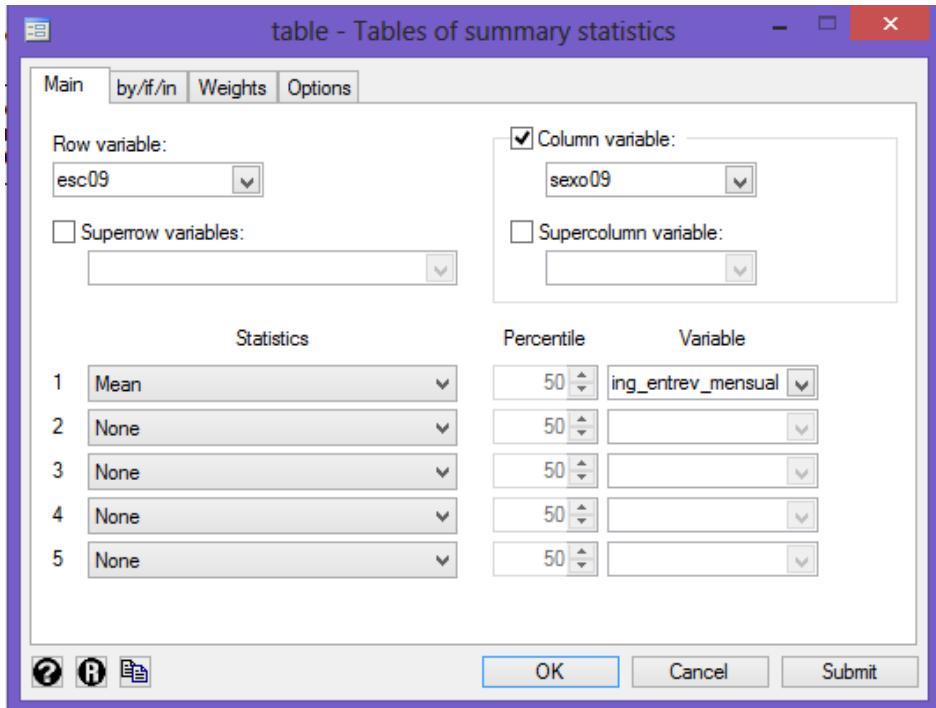
```
Summary for variables: ing_entrev_mensual
by categories of: sexo09 (a8. sexo)
```

sexo09	mean	p50	min	max
1	304912.1	195566.7	0	2.04e+07
2	158044.8	82000	0	1.50e+07
Total	229710.1	140000	0	2.04e+07

Como hemos visto en los casos anteriores también podemos tabular usando las ventanas:



Si quisiéramos graficar escolaridad contra sexo hacemos doble click sobre "Table of summary statistics" observamos el dibujo que viene y ponemos "esc" en "Row variable" y "sexo" sobre "Column variable".



Capítulo V: Modificación de una base de datos.

En esta sección veremos como se pueden modificar las bases de datos, desde los nombres de sus variables y la generación de nuevas variables hasta la forma se como se pueden unir dos bases de datos

Modificación de Variables de una base de datos

Para lograr que una base de datos sea más amigable y sea entendida por cualquier usuario, es recomendable incorporar etiquetas a los números o códigos de las variables. Esto se hace mediante la utilización de variables secundarias llamadas **value labels**. Si una variable tiene una variable secundaria que entregue etiqueta a los códigos que contiene, debería aparecer en el resultado del comando describe.

Por ejemplo, la variable sex09:

```
. describe sexo09
```

variable name	storage type	display format	value label	variable label
sexo09	byte	%9.0g		a8. sexo
.				

No tiene asociada una variable secundaria que nos indique que significa el número 1 y el número 2 en esta variable.

Entonces para etiquetar los códigos de una variable se deben realizar dos pasos:

Crear la variable secundaria (value label) que realice el nexo entre los códigos y sus etiquetas:

```
. label define sexolbl 1 "Hombre" 2 "Mujer"
```

Indicar que la relación entre la variable y su variable secundaria con las etiquetas:

```
. label values sexo09 sexolbl
```

Al hacer un **describe** de la variable “sexo09”, este indica que existe una variable que contiene la etiqueta de los números o códigos de esta variable:

```
. describe sexo09
```

variable name	storage type	display format	value label	variable label
sexo09	byte	%9.0g	sexolbl	a8. sexo

	folio	edad09	esc09	sexo09	
1	2409302	19	12	2	
2	2403092	19	13	2	
3	2405087	20	13	1	
4	2400943	20	12	2	
5	2403511	21	16	1	
6	2402495	21	12	2	
7	2418471	21	16	2	
8	2406608	21	12	1	
9	2406586	21	12	2	
10	1280487	22	12	1	
11	2416135	22	12	2	
12	2400922	22	16	1	

	folio	edad09	esc09	sexo09	
1	2409302	19	12	Mujer	
2	2403092	19	13	Mujer	
3	2405087	20	13	Hombre	
4	2400943	20	12	Mujer	
5	2403511	21	16	Hombre	
6	2402495	21	12	Mujer	
7	2418471	21	16	Mujer	
8	2406608	21	12	Hombre	
9	2406586	21	12	Mujer	
10	1280487	22	12	Hombre	
11	2416135	22	12	Mujer	
12	2400922	22	16	Hombre	

Si es que la base de datos que uno tiene ya viene con las variables secundarias de etiquetas (value labels), y queremos saber que código esta relacionado a que etiqueta, se debe utilizar el comando **label list**:

```
. label list sexolbl
```

```
sexolbl:
```

```
  1 Hombre
  2 Mujer
```

Por otro lado si queremos en ese instante cambiar el nombre de alguna variable y su respectiva descripción, podemos utilizar el administrador de variables (variables manager)

The screenshot shows the SPSS Variables Manager dialog box. On the left, the 'Variables' tab is selected, displaying a list of variables with their properties. The variable 'sexo09' is currently selected. On the right, the 'Variable Properties' panel is open, showing detailed settings for 'sexo09': Name (sexo09), Label (a8_sexo), Type (byte), Format (%9.0g), Value Label (sexolbl), and Notes (empty). A 'Create...' button is available for the Format field.

#	Variable	Label	Type	Format	Value Label	Notes
	folio		double	%9.0g		
	edad09	a9. ¿qué edad tiene ud.?	int	%9.0g		
	esc09	Años de escolaridad EPS09	float	%9.0g		
	sexo09	a8_sexo	byte	%9.0g	sexolbl	
	b2_09	Estatus laboral Abril 2009	float	%32.0g	b2	
	b4_09	Region en que trabaja (Abril...)	float	%32.0g	b4	
	b6_09	Tipo de trabajo (Abril 2009)	float	%32.0g	b6	
	b8_09	Categoría Ocupacional (Abril...)	float	%32.0g	b8	
	b9a_09	Tiene contrato (Abril 2009)	float	%9.0g	b9a	
	b9b_09	Relación Contractual (Abril ...)	float	%32.0g	b9b	
	b10_09	Horario (Abril 2009)	float	%40.0g	b10	

En **Name** escribimos el nombre de la variable y en **Label** su descripción. También es posible cambiar el nombre de la variable con el comando **rename**:

```
. rename edad09 edad
```

y cambiar la descripción de la variable con el comando **label variable**:

```
| . label variable folio "Identificador individual"
```

Cuando una variable está en formato *string* (no numérico) no se pueden obtener estadísticas de ella. Los comandos **encode** y **decode** cambian el formato de una variable string a numérico y viceversa:

```
encode variable, generate(nueva_variable)
```

Mediante este código generamos una nueva variable que le asigna un código a cada palabra distinta en la variable y deja como etiqueta la palabra.

Ahora cuando la variable que no tiene formato numérico, pero en realidad es un número sólo que al traspasarlo a la base de datos quedó en formato string, lo recomendable es utilizar el siguiente comando:

```
generate nueva_variable=real(variable)
```

Por ejemplo, disponemos de la siguiente base de datos (ejstring.dta):

```
. use Bases/ejstring.dta, clear
```

	país	var1	var2
1	Argentina	11.3	1500
2	Argentina		1350
3	Argentina	12.5	1400
4	Argentina	11.5	2000
5	Argentina	10	2100
6	Bolivia	20	3000
7	Bolivia	20.3	4000
8	Bolivia	22.5	2400

Las variable país y var1 tienen formato string, en el primer caso es una variable que en esencia es no numérica pero podríamos querer codificar, en el segundo caso es una variable que debería ser numérica, no queremos codificar, sino que se transforme en formato numérico:

```
. encode país, generate(pais2)
```

	país	var1	var2	pais2
1	Argentina	11.3	1500	Argentina
2	Argentina		1350	Argentina
3	Argentina	12.5	1400	Argentina
4	Argentina	11.5	2000	Argentina
5	Argentina	10	2100	Argentina
6	Bolivia	20	3000	Bolivia
7	Bolivia	20.3	4000	Bolivia
8	Bolivia	22.5	2400	Bolivia

```
. g var1_2=real(var1)
(1 missing value generated)
```

Data Editor (Edit) - [ejstrin]

File Edit Data Tools

var8[13]

	pais	var1	var2	pais2	var1_2
1	Argentina	11.3	1500	Argentina	11.3
2	Argentina		1350	Argentina	.
3	Argentina	12.5	1400	Argentina	12.5
4	Argentina	11.5	2000	Argentina	11.5
5	Argentina	10	2100	Argentina	10
6	Bolivia	20	3000	Bolivia	20
7	Bolivia	20.3	4000	Bolivia	20.3
8	Bolivia	22.5	2400	Bolivia	22.5

Unir bases de datos: merge y append

Muchas veces es necesario combinar dos o más bases de datos para formar una sola. Para ello se pueden utilizar los comandos `merge` y `append`. El primero une dos bases de datos utilizando una variable en común (generalmente es un folio o código que identifica las observaciones en la base de datos). Las dos bases de datos deben estar guardadas en formato .dta y deben estar ordenadas de acuerdo a la variable que se va a pegar. El objetivo del comando `merge` es agregar variables (columnas) no observaciones (filas). Supongamos que tenemos dos bases de datos, la primera llamada uno.dta contiene la siguiente información:

id	esc	sexo	edad
1	8	1	20
2	9	2	55
3	11	2	45
4	13	2	47

5	15	1	32
6	2	2	31
7	0	1	28
8	10	1	26
9	9	1	35

Y supongamos que tenemos otra base de datos (dos.dta) con la siguiente información:

id	ingreso
1	80000
3	99000
4	110000
6	130000
8	150000
9	200000
10	115000
11	98000

Entonces si queremos pegar (match-merge) la información de ambas bases de datos, para formar una sólo base de datos que contenga las variables: id, esc, sexo, edad e ingreso, debemos realizar el siguiente procedimiento:

Ordenar la base de datos dos.dta (using data set) de acuerdo a las variables que vamos a hacer el pegado (id) y guardar esta base de datos:

sort	id
------	----

```
save dos.dta, replace
```

Abrir la base de datos uno.dta (master data set), ordenarla de acuerdo a id y pegar la información de la base dos.dta:

```
use uno.dta  
sort id  
merge id using dos.dta
```

La base de datos que resulta de esto es de la siguiente forma:

id	esc	sexo	edad	ingreso	_merge
1	8	1	20	80000	3
2	9	2	55	.	1
3	11	2	45	99000	3
4	13	2	47	110000	3
5	15	1	32	.	1
6	2	2	31	130000	3
7	0	1	28	.	1
8	10	1	26	150000	3
9	9	1	35	200000	3
10	.	.	.	115000	2
11	.	.	.	98000	2

Al realizar este pague de datos, el programa generará una variable, si Ud. no le asigna un nombre se creará con el nombre _merge, pero Ud. puede darle el nombre que desee de la siguiente forma:

```
merge id using dos.dta, _merge(nombre)
```

Esta variable puede tomar tres valores:

_merge=1 cuando la observación esta **sólo** en Master Dataset
_merge=2 cuando la observación esta **sólo** en Using Dataset

Veamos un ejemplo con la base de datos expeps09.dta, en su carpeta posee otra base de datos llamada expeps06.dta que corresponde a la misma encuesta pero realizada en el 2006 para los mismos entrevistados (parte de ellos)

Si quiero construir entonces una sola base de datos que contenga todas estas variables en forma conjunta, debería realizar los siguientes pasos:

```
. use Bases/expeps06.dta
. sort folio
. save Bases/expeps06.dta, replace
file Bases/expeps06.dta saved
. use Bases/expeps09.dta
. sort folio
. merge folio using Bases/expeps06.dta
(note: you are using old merge syntax; see \[R\] merge for new syntax)
```

Lo que este pequeño código nos indica es que carguemos la base de datos exteps06.dta, la ordenemos (utilizando el comando `sort`) de acuerdo al identificador “folio”, y guardemos los cambios realizados (esta base de datos se denomina `using dataset`). Luego abrimos la base de datos a la cual le vamos a pegar variables (`master dataset`), exteps09.dta, la ordenamos de acuerdo al identificador folio, y finalmente le pegamos la información que esta en la base exteps06.dta utilizando el comando `merge`.

Si alguna de las bases de datos no esta ordenada les arrojara un mensaje de error:

```
using data not sorted
master data not sorted
```

El resultado de pegar ambas bases de datos se puede revisar con la variable `_merge` que se genera de manera automática:

<code>_merge</code>	Freq.	Percent	Cum.
1	1,059	6.05	6.05
2	3,259	18.62	24.67
3	13,184	75.33	100.00
Total	17,502	100.00	

La nueva versión del comando es la siguiente (no requiere ordenar las bases de datos):

<code>. use Bases/exteps09.dta, clear</code>	<code>. merge 1:1 folio using Bases/exteps06.dta</code>
	<code>Result # of obs.</code>
	<code>not matched 4,318</code>
	<code>from master 1,059 (_merge==1)</code>
	<code>from using 3,259 (_merge==2)</code>
	<code>matched 13,184 (_merge==3)</code>

Por otro lado, el comando **append**, anexa observaciones para el mismo conjunto de variables, es decir, agrega filas a la base de datos. Supongamos que además de uno.dta tenemos otra base de datos (tres.dta) que contiene las mismas variables que la primera pero para otros 10 individuos distintos:

id	esc	sexo	edad
10	8	1	20
11	9	2	55
12	11	2	45
13	13	2	47
14	15	1	32
15	2	2	31
16	0	1	28
17	10	1	26
18	9	1	35
19	4	2	20

Entonces podemos juntar estas 19 observaciones en una sola base de datos mediante el comando **append**, de la siguiente forma:

```
use dos.dta  
append using tres.dta
```

El resultado es el siguiente:

id	esc	sexo	edad
1	8	1	20
2	9	2	55
3	11	2	45
4	13	2	47
5	15	1	32
6	2	2	31
7	0	1	28
8	10	1	26
9	9	1	35
10	8	1	20
11	9	2	55
12	11	2	45
13	13	2	47
14	15	1	32
15	2	2	31
16	0	1	28
17	10	1	26
18	9	1	35
19	4	2	20

Supongamos ahora que tenemos información laboral mensual recolectada en la encuesta EPS 2009:

	folio	fecha	b2	b4	oficio	b6
40	179	200904	inactivo	.	.	.
41	717	200601	trabajan	15 regió	7	permanen
42	717	200602	trabajan	15 regió	7	permanen
43	717	200603	trabajan	15 regió	7	permanen
44	717	200604	trabajan	15 regió	7	permanen
45	717	200605	trabajan	15 regió	7	permanen
46	717	200606	trabajan	15 regió	7	permanen
47	717	200607	trabajan	15 regió	7	permanen
48	717	200608	trabajan	15 regió	7	permanen
49	717	200609	trabajan	15 regió	7	permanen
50	717	200610	trabajan	15 regió	7	permanen
51	717	200611	trabajan	15 regió	7	permanen
52	717	200612	trabajan	15 regió	7	permanen
53	717	200701	trabajan	15 regió	7	permanen
54	717	200702	trabajan	15 regió	7	permanen
55	717	200703	trabajan	15 regió	7	permanen
56	717	200704	trabajan	15 regió	7	permanen
57	717	200705	trabajan	15 regió	7	permanen
58	717	200706	trabajan	15 regió	7	permanen

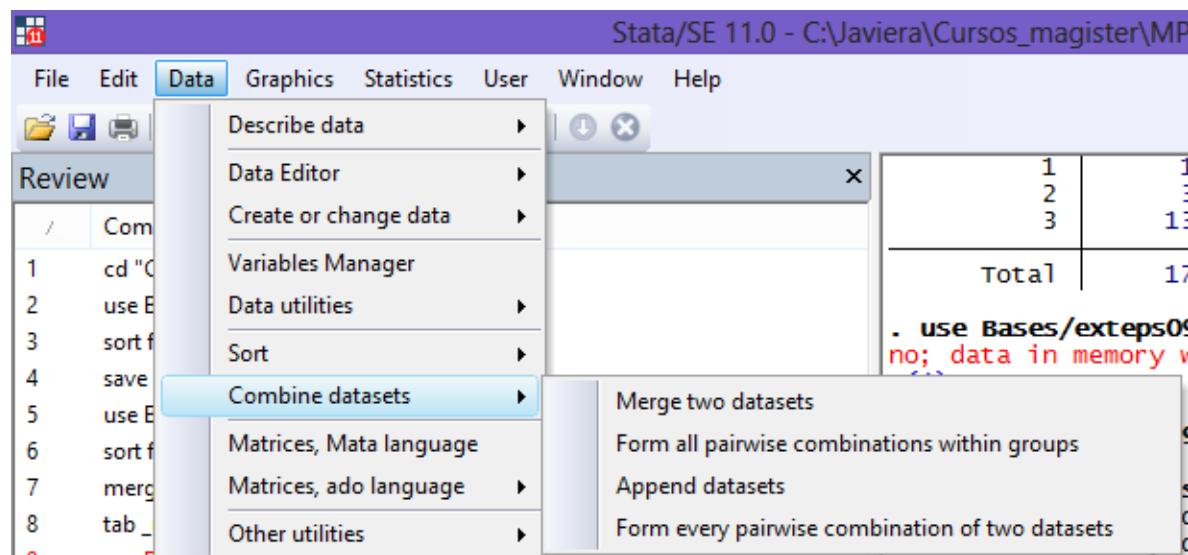
Y lo mismo proveniente de la encuesta EPS 2006:

	folio	fecha	b2	b6	b8	oficio
94	547	200611	4	.	.	.
95	547	200612	4	.	.	.
96	547	200701	4	.	.	.
97	547	200702	1	1	3	2
98	547	200703	1	1	3	2
99	547	200704	1	1	3	2
100	547	200705	1	1	3	2
101	547	200706	1	1	3	2
102	547	200707	1	1	3	2
103	547	200708	1	1	3	2
104	717	200201	1	2	2	9
105	717	200202	1	2	2	9
106	717	200203	1	2	2	9
107	717	200204	1	2	2	9
108	717	200205	1	2	2	9
109	717	200206	1	2	2	9
110	717	200207	1	2	2	9
111	717	200208	1	2	2	9

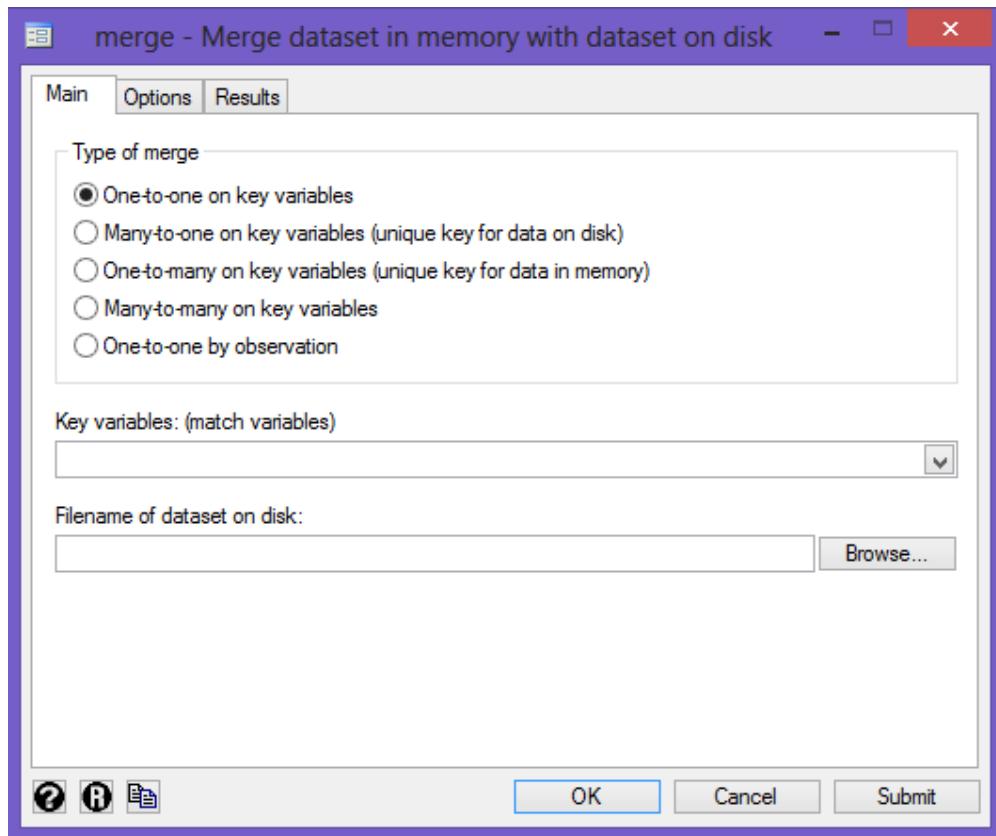
Podemos juntar la información de ambas bases en una sola de la siguiente forma:

```
. use Bases/ext_mensualeps06.dta, clear  
. append using Bases/ext_mensualeps09.dta
```

Podemos juntar bases de datos usando las ventanas también; en la opción "Data" existe la opción "Combine datasets" luego se hace doble clic en "Merge" o "Append" y nos sale una ventana que nos permite juntar dos o múltiples bases de datos:



Luego, para el caso de `merge`, buscamos la base que le queremos añadir a la base que estamos usando y ponemos su dirección donde sale "Filename of dataset on disk", luego en "Key variables" ponemos el nombre de la variable mediante el cual queremos unir las dos bases de datos.



Condensar una base de datos: `collapse`

En algunas ocasiones las bases de datos disponen de más de una observación por individuo, país, empresa, etc. Si nos interesa trabajar sólo con una observación por individuo podemos condensar la base de datos mediante el comando `collapse`.

Suponga que tiene una base de datos de hogares y que cada hogar tiene una observación para cada miembro del hogar que lo integra. Si cada hogar dispone de un identificador único, entonces se puede formar una base de datos alternativa que contenga una sola observación por hogar (en lugar de una observación por individuo) para cada una de las variables deseadas. Esta observación puede contener la media, desviación estándar, suma, u otro estadístico por hogar (sino se especifica calcula la media). Por ejemplo:

```
collapse (mean) edad (max) educacion (p50) ingreso, by(hogar)
```

El código anterior crea una base de datos con cuatro variables (hogar, edad, educación e ingreso) con una observación por hogar, la cual contiene el promedio de la edad por hogar, el máximo de la educación por hogar y la mediana del ingreso por hogar.

Cambiar la estructura de la base de datos: reshape

Para explicar como se utiliza este comando primero vamos a definir dos representaciones de las bases de datos: **wide form** (forma horizontal) y **long form** (forma vertical).

Long form:

	Empresa	año	precio
1	1	2007	100
2	1	2008	110
3	1	2009	120
4	2	2007	101
5	2	2008	132
6	2	2009	124
7	3	2008	120
8	3	2009	118
9	4	2007	119
10	4	2008	111
11	4	2009	123
12	5	2007	132
13	5	2008	142

Wide form:

	Empresa	precio2007	precio2008	precio2009
1	1	100	110	120
2	2	101	132	124
3	3	.	120	118
4	4	119	111	123
5	5	132	142	110
6	6	123	125	132
7	7	.	133	135
8	8	.	145	.
9	9	.	151	.
10	10	.	161	.

En la base de datos en forma vertical (long form) podemos ver que existe una variable que es constante al interior de un “grupo”, en este caso, al interior de una empresa, tenemos una variable que varía al interior de este grupo, el precio, y una variable que me sirve para identificar las distintas observaciones al interior de un grupo, que es la variable “año”.

En la base de datos en forma horizontal (wide form), tengo una sola observación por empresa pero tengo más de una variable precio, una para cada año.

El comando `reshape` me permite intercambiar las bases de datos entre estos dos tipos de formatos, de la siguiente forma:

```
reshape long precio, i(Empresa) j(año) (Wide →Long)
```

Long → Wide

	Empresa	año	precio
1	1	2007	100
2	1	2008	110
3	1	2009	120
4	2	2007	101
5	2	2008	132
6	2	2009	124
7	3	2008	120
8	3	2009	118
9	4	2007	119
10	4	2008	111
11	4	2009	123
12	5	2007	132

```
. use Bases/ejreshape.dta, clear
. reshape wide precio, i(Empresa) j(año)
(note: j = 2007 2008 2009)

Data                                long    ->    wide
Number of obs.                      22    ->    10
Number of variables                 3    ->    4
j variable (3 values)               año   ->    (dropped)
xij variables:                      precio  ->    precio2007  precio2008  precio2009
```

	Empresa	precio2007	precio2008	precio2009
1	1	100	110	120
2	2	101	132	124
3	3	.	120	118
4	4	119	111	123
5	5	132	142	110
6	6	123	125	132
7	7	.	133	135
8	8	.	145	.
9	9	.	151	.
10	10	.	161	.

Wide → Long

	Empresa	precio2007	precio2008	precio2009
1	1	100	110	120
2	2	101	132	124
3	3	.	120	118
4	4	119	111	123
5	5	132	142	110
6	6	123	125	132
7	7	.	133	135
8	8	.	145	.
9	9	.	151	.
10	10	.	161	.

```
. reshape long precio, i(Empresa) j(año)
(note: j = 2007 2008 2009)
```

Data	wide	->	long
Number of obs.	10	->	30
Number of variables	4	->	3
j variable (3 values)		->	año
xij variables:	precio2007	precio2008	precio2009
			-> precio

	Empresa	año	precio
1	1	2007	100
2	1	2008	110
3	1	2009	120
4	2	2007	101
5	2	2008	132
6	2	2009	124
7	3	2008	120
8	3	2009	118
9	4	2007	119
10	4	2008	111
11	4	2009	123
12	5	2007	132

Eliminar variables: keep y drop

Ahora, si se tiene una base de datos que contiene muchas variables y sólo se va a trabajar con algunas de ellas, se puede utilizar el comando **keep** para mantener en la base de datos sólo las variables que interesan. Por ejemplo:

```
keep esc ingreso
```

También podríamos requerir mantener todas las variables pero sólo un subconjunto de las observaciones, como por ejemplo, las de los hombres:

```
keep if dhombre==1
```

o las de las personas con menos de 12 años de escolaridad

```
keep if esc<=12
```

o las de las personas con 200.000 pesos o más de ingreso

```
keep if ingreso>=200000
```

El comando **drop** cumple el mismo rol que el comando **keep**, sin embargo, éste borra observaciones o variables, en vez de mantenerlas. Para realizar lo mismo que hicimos con el comando **keep** pero utilizando **drop**, se debería hacer lo siguiente:

```
drop expr dhombre folio  
drop if dhombre==0  
drop if esc>12  
drop if ingreso<200000
```

Usando ventanas también podemos aplicar el comando **keep** y **drop** a través de la ventada de administración de variables.

Crear variables: generate y egen

Uno de los comandos más relevantes en STATA es **generate ("gen" o "g")**. Este comando genera una nueva variable definida en base a una expresión numérica, la cual puede contener otras variables.

El siguiente cuadro resume las expresiones más utilizadas:

+	suma	\geq	Mayor o igual	&	y
-	resta	$>$	Mayor estricto que		o
*	multiplicación	\leq	Menor o igual que	exp()	exponencial
/	división	$<$	Menor estricto que	log()	logarítmico
= =	igual	\neq	Distinto que	sum()	suma

Ejemplos:

```
generate age2 = age*age  (genera variable edad al cuadrado)  
gen unitpr = cost/quantity  
  
gen xlag = x[_n-1]
```

Otro ejemplo. En la base `ingreso.dta` se tienen los años de escolaridad (`esc`); a partir de esta variable se podría generar una nueva (`tesc`) con el nivel de educación alcanzado por cada individuo (Básica Incompleta, Básica Completa, Media Incompleta, etc..). Esto se hace con el siguiente código:

```
g tesc=.  
replace tesc=1 if esc<8  
replace tesc=2 if esc==8  
replace tesc=3 if esc>8 & esc<12  
replace tesc=4 if esc==12  
replace tesc=5 if esc>12
```

El comando `replace`, reemplaza observaciones que cumplen con cierta condición con el valor que uno especifica.

Otro comando útil, similar a `generate`, es el comando `egen`. Este contiene una gran cantidad de funciones pre-establecidas con las que se pueden generar nuevas variables. La diferencia con el comando `generate` es que en este uno define como quiere generar la variable (sumando, restando, multiplicando, etc.), sin embargo, el comando `egen` tiene funciones ya establecidas, por ejemplo, el promedio, la desviación estándar, etc...

Algunos ejemplos de estas funciones son:

count(x): crea una variable que contiene el numero de observaciones (distintas de ·) en x.

concat(x y ... z): concatena las variables entre paréntesis, genera una variable con formato string (no numérico)

group(x y ... z): genera una variable que tomas los valores 1,2,... para los grupos formados por las variables entre paréntesis. El orden de los grupos va a depender de cómo estén ordenadas las variables entre paréntesis.

max(x): genera una variable que contiene el máximo valor de x

mean(x): genera una variable que contiene el promedio de x

median(x): genera una variable que contiene la mediana de x

min(x): genera una variable que contiene el mínimo valor de x

rmax(x y ... z): entrega el máximo valor entre x, y, ..., z para cada observación (fila)

rmean(x y ... z): entrega el promedio de x, y, ..., z para cada observación (fila)

rmin(x y ... z): entrega el mínimo valor entre x, y, ..., z para cada observación (fila)

rmiss(x y ... z): entrega el número de missing values (sin observación) en x, y, ..., z para cada observación (fila)

rsum(x y ... z): entrega la suma de x, y, ..., z para cada observación (fila)

seq(), from(a) to(b) block(c): genera una secuencia de números de a hasta b en bloques de c.

sum(x): genera una variable que contiene la suma de x.

La mayoría de estas funciones pueden ser combinadas con el comando **by**, el que permite generar variables por diversas categorías. Por ejemplo, si se quiere crear el promedio de la experiencia laboral para cada año de educación, se debe realizar lo siguiente:

```
egen promexpr=mean(expr), by(esc)
```

```
by esc: egen promexpr=mean(expr)
```

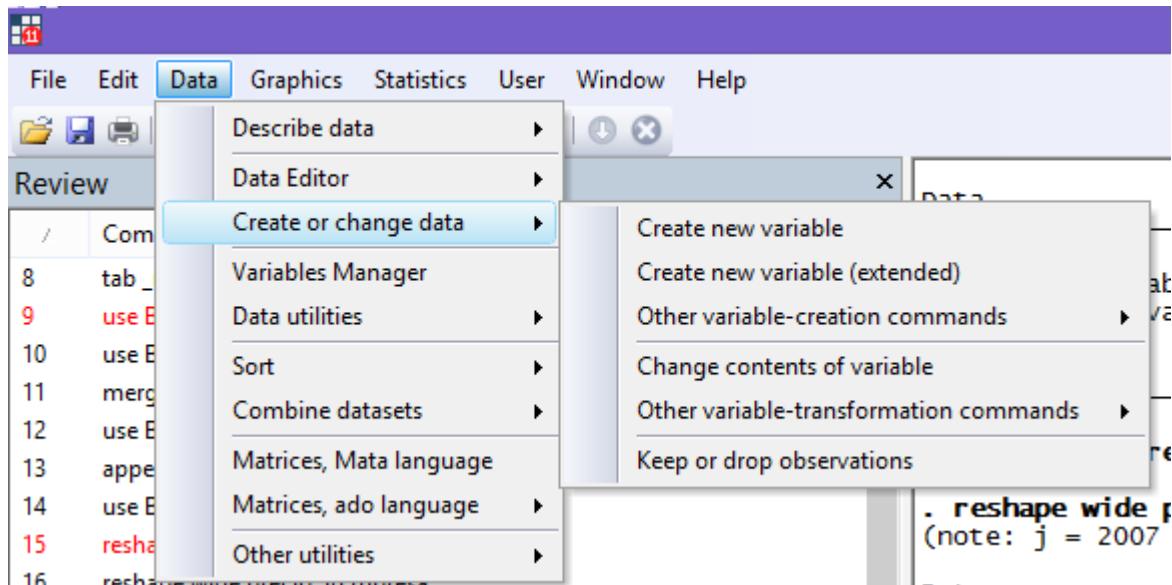
La segunda forma requiere que la base de datos esté ordenada de acuerdo a la variable utilizada en **by**, en este caso, debemos tipar antes:

```
sort esc
```

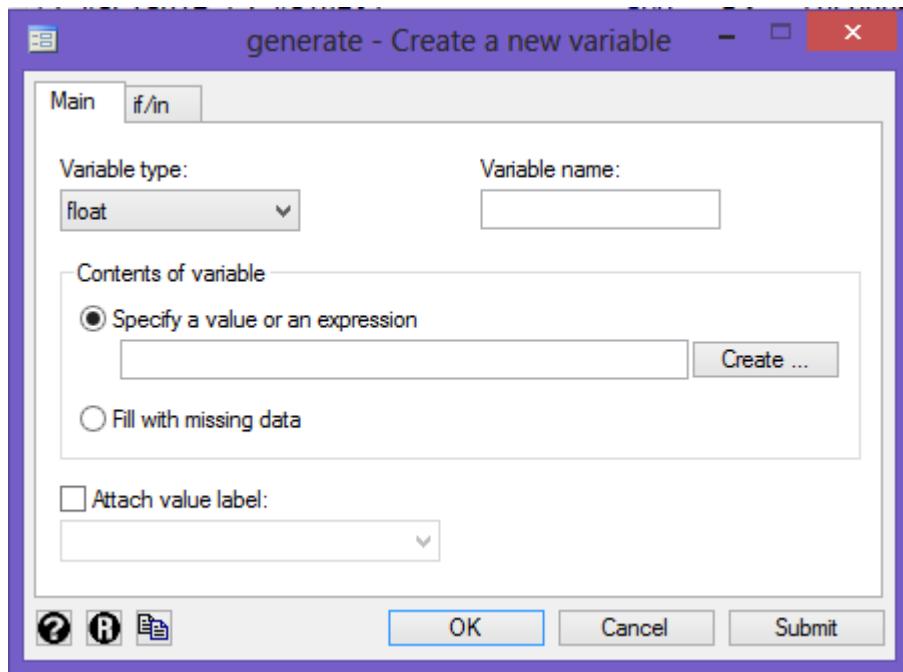


En esta parte veremos de manera más aplicada los comandos utilizados en las clases anteriores; para ello será necesario enseñar a usar los archivos do y los archivos log, que como veremos serán de gran utilidad, ya que nos permitirán ir guardando información. Finalmente veremos el comando help; el cual es un comando auxiliar que nos permite encontrar los comandos necesarios para realizar un gama de funciones en stata.

Como con casi todos los comandos también se pueden obtener los mismos resultados, mediante el uso de ventanas:



Luego de apretar doble clic sobre "Create new variable" podemos ver que es trivial la forma de generar otra variable.



Capítulo VI: Aplicando lo aprendido I

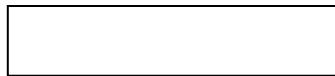
En esta sección veremos de manera aplicada lo que hemos aprendido hasta ahora y en los dos siguientes mostraremos a partir de esta sección la utilidad de los archivos do y log. El archivo log es el archivo donde se va guardando todo lo que aparece en la ventana de resultados, mientras que el do es un archivo que nos permite programar todo el trabajo que queremos realizar en stata.

Un ejemplo aplicado

Supongamos que, utilizando la encuesta casen 2009, queremos obtener la tasa de pobreza y la tasa de indigencia; además el número de personas que caen en ésta categoría.

El primer paso es establecer memoria (si no está preestablecida en el programa) y luego abrir la base de datos:

```
set mem 700m  
use Casen2009.dta
```



Luego debemos empezar a generar las variables necesarias para encontrar las tasas de pobreza e indigencia correspondiente. Como la encuesta CASEN se realiza a hogares y nosotros estamos interesados en el porcentaje de individuos pobres e indigentes y no en el porcentaje de hogares pobres e indigentes; lo que necesitamos es tener algún indicador per cápita.

Lo que generaremos a continuación es una variable que generará un identificador que será el mismo para los individuos que pertenecen a un mismo. La variable o indica el orden de la persona dentro del hogar, por lo cual la combinación de las variables identificador del hogar (que vamos a generar) y o deberían ser únicas.

```
. use Bases/casen2009.dta, clear  
. egen idhogar=group(segmento folio)  
. duplicates report idhogar o  
Duplicates in terms of idhogar o
```

copies	observations	surplus
1	246924	0

```
. egen np=count(idhogar), by(idhogar)  
. order idhogar o np  
. edit idhogar o np
```

Data Editor

File Edit Data Tools

var353[19]

Snapshots

	idhogar	o	np
1	1	1	3
2	1	2	3
3	1	3	3
4	2	1	4
5	2	2	4
6	2	3	4
7	2	4	4
8	3	1	2
9	3	2	2
10	4	1	5
11	4	2	5
12	4	3	5
13	4	4	5
14	4	5	5

. sum idhogar

Variable	obs	Mean	Std. Dev.	Min	Max
idhogar	246924	35770.24	20797.24	1	71460

En la tabla anterior podemos ver que hay 71,460 hogares y 246,924 individuos; además podemos ver que el número de individuos por hogar varía. La variable idhogar tiene asignado un código por hogar, vemos por ejemplo que el hogar 4 tiene asociados 5 individuos, y a su vez el hogar número 3 tiene asociados 2 individuos.

Ahora, sabiendo el número de personas que viven en el hogar, generaremos el ingreso per cápita del hogar para lo cual tenemos que dividir el ingreso total del hogar por el número de personas que viven en él. Veamos los diferentes tipos de personas (relación de parentesco con el jefe de hogar) que viven en el hogar:

r1: parentesco con el jefe(a) de hogar	Freq.	Percent	Cum.
jefe(a) de hogar	71,460	28.94	28.94
esposo(a)/pareja	46,609	18.88	47.82
hijo(a) de ambos	64,683	26.20	74.01
hijo(a) sólo del jefe	24,957	10.11	84.12
hijo(a) sólo del esposo(a)/pareja	3,464	1.40	85.52
padre o madre	2,037	0.82	86.35
suegro(a)	1,047	0.42	86.77
yerno o nuera	3,615	1.46	88.23
nieto(a)	18,861	7.64	95.87
hermano(a)	2,816	1.14	97.01
cuñado(a)	812	0.33	97.34
otro familiar	4,710	1.91	99.25
no familiar	1,711	0.69	99.94
servicio doméstico puertas adentro	142	0.06	100.00
Total	246,924	100.00	

Como vemos en algunas casas ocurre que hay personas que prestan servicios domésticos puertas adentro y generalmente para calcular el ingreso per cápita del hogar debemos excluir a esas personas. Para ello generaremos una variable llamada `s`; la cual tomará el valor 1 si la personas pertenece al hogar y no presta servicios dentro de él; y toma el valor de 0 si la personas presta servicios dentro del hogar. Después generaremos el número de personas que pertenecen al hogar y no prestan servicios dentro de él. Finalmente botamos la variable `s`, ya que la creamos sólo como una variable momentánea para excluir a las personas que prestan servicios domésticos puertas adentro y obtendremos la variable `nps`, que es la que nos señala el número de personas que realmente son parte del hogar.

Primero para saber que código de la variable `pco1` corresponde a servicio doméstico:

```

. d pco1
      variable name  storage  display   value
                           type    format   label
                                         variable label
pco1          byte     %8.0g    pco1    r1: parentesco con el jefe(a) de hogar
. label list pco1
pco1:
  1 jefe(a) de hogar
  2 esposo(a)/pareja
  3 hijo(a) de ambos
  4 hijo(a) sólo del jefe
  5 hijo(a) sólo del esposo(a)pareja
  6 padre o madre
  7 suegro(a)
  8 yerno o nuera
  9 nieto(a)
 10 hermano(a)
 11 cuñado(a)
 12 otro familiar
 13 no familiar
 14 servicio doméstico puertas adentro

```

```

. g s=1
. replace s=0 if pco1==14
(142 real changes made)
. egen nps=sum(s), by(idhogar)
. drop s

```

Ahora, que tenemos el número de personas que pertenecen al hogar, estamos listos para crear la variable que nos indica el nivel de ingreso individual de los habitantes del país y poder separarlo de acuerdo al nivel de pobreza.

En el año 2009, se define a una persona como pobre si está recibe ingresos menores a \$64,134 y vive en zona urbana y \$43,242 y está vive en zona rural. Una persona será indigente si recibe ingresos menores a \$32,067 y vive en zona urbana; y \$24,710 si esta vive en zona rural. Como se está pidiendo el número y le porcentaje de pobres e indigentes, necesitamos clasificar a los individuos en alguna de las tres categorías posible; no pobres; pobres no indigentes e indigentes.

```

. g ing=ytotthaj/nps
. g pobre=1 if ing<32067& z==1
(239914 missing values generated)
. replace pobre=1 if ing<24710 & z==2
(3888 real changes made)
. replace pobre=2 if ing>=32067 & ing<64134 & z==1
(22955 real changes made)
. replace pobre=2 if ing>=24710 & ing<43242 & z==2
(7407 real changes made)
. replace pobre=3 if pobre=.
(205664 real changes made)
. replace pobre=. if pcol1==14
(142 real changes made, 142 to missing)

```

Una vez generada la variable podemos etiquetarla con una descripción que nos ayude a recordar que información contiene esta variable, como también podemos etiquetar las categorías de la variable.

```

. label variable pobre "Situacion de pobreza"
. label define pobrelbl 1 "Indigente" 2 "Pobre no indigente" 3 "No pobre"
. label values pobre pobrelbl
. tab pobre

```

Situacion de pobreza	Freq.	Percent	Cum.
Indigente	10,893	4.41	4.41
Pobre no indigente	30,362	12.30	16.72
No pobre	205,527	83.28	100.00
Total	246,782	100.00	

Finalmente debemos realizar la tabulación de tal forma de captar no sólo el número de personas pobres e indigentes, sino que también su porcentaje. Es importante notar que hay que usar factor de expansión; ya que ello nos entregará el valor poblacional de los números que buscamos:

```
. tab z_pobre [w=expr], col  
(frequency weights assumed)
```

Key
<i>frequency column percentage</i>

zona	Situacion de pobreza			Total
	Indigente	Pobre no	No pobre	
urbano	537,322 84.71	1,743,254 90.34	12,468,471 86.65	14,749,047 87.00
rural	97,006 15.29	186,450 9.66	1,920,309 13.35	2,203,765 13.00
Total	634,328 100.00	1,929,704 100.00	14,388,780 100.00	16,952,812 100.00

Archivos log

Existen dos formas de trabajar en Stata: interactiva y programada. Cuando se trabaja en forma programada se escribe en un archivo **do** todos los comandos que se quieren ejecutar y luego se corre el programa de una vez para obtener los resultados, esto será visto en la otra sección. Cuando se trabaja en forma interactiva, se ejecutan los comandos directamente en la ventana **Command**, viendo los resultados en la ventana **Results**. Cuando se trabaja de esta forma es importante ir registrando todos los pasos realizados, en caso que se quiera repetir algo en una siguiente ocasión o simplemente para tener un respaldo de todo lo realizado. Para esto se utilizan los archivos **log**, estos no son más que un archivo de texto que contiene todo los comandos ejecutados y sus respectivos resultados.

El archivo log del ejercicio visto anteriormente:

Viewer (#1) [view "logclase1.log"]

Advice Contents What's New News

```
name: <unnamed>
log: C:\Javier\cursos_magister\MPP\Nivelacion de Stata\2013\logclase1.log
log type: text
opened on: 13 Mar 2013, 16:21:07

. use Bases/casen2009.dta, clear
. egen idhogar=group(segmento folio)
. duplicates report idhogar o
Duplicates in terms of idhogar o

-----+-----+-----+
      copies | observations      surplus
-----+-----+
      1 |      246924          0
-----+-----+-----+-----+-----+-----+
. egen np=count(idhogar), by(idhogar)
. order idhogar o np
. edit idhogar o np
. sum idhogar
      variable |       obs        mean     std. dev.      min      max
idhogar |    246924    35770.24    20797.24         1      71460
.
```

Siempre es recomendable abrir un archivo log al inicio de cualquier trabajo:

log using clase1.log

Si el archivo log que estamos tratando de abrir ya existía, el programa nos entregará el siguiente error:

file C:\Nivelacion_Stata\clase1.log already exists

Frente a esto tenemos dos opciones, cual de ellas se tome depende de los objetivos: reemplazar el archivo ya existente, o seguir escribiendo en el archivo existente a continuación de lo último ingresado.

Para reemplazar el archivo existente:

```
log using clase1.log, replace
```

Para seguir a continuación del archivo existente:

```
log using clase1.log, append
```

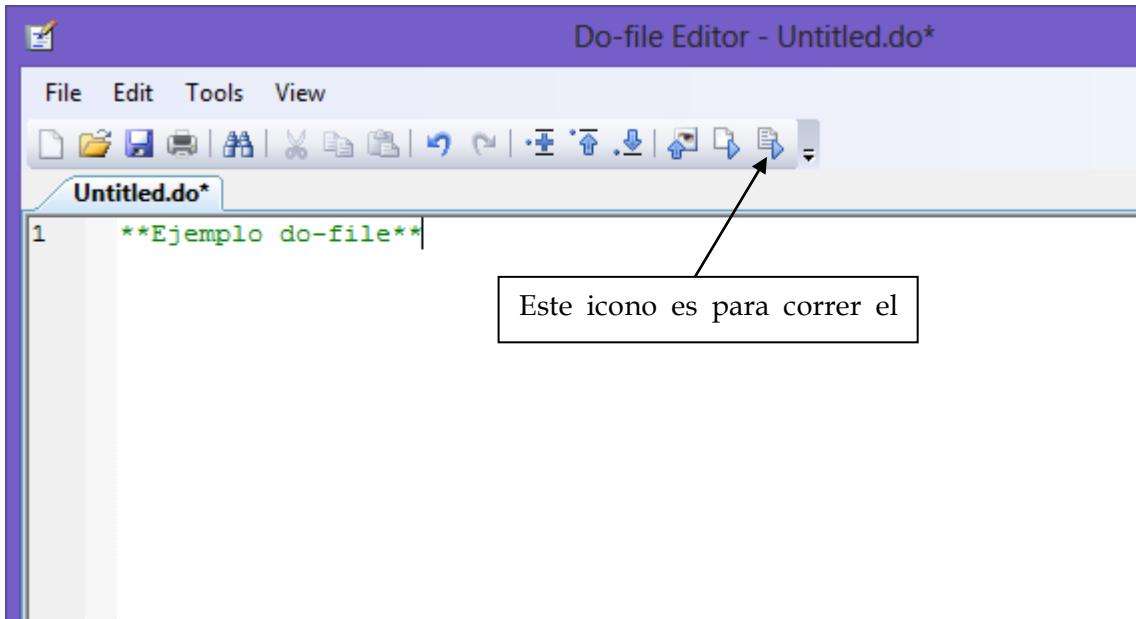
Archivos do: para trabajar en forma programada en STATA

Como se ha mencionado en clases anteriores, existen dos formas de trabajar en STATA, en forma interactiva y en forma programada. La primera forma consiste en ir ejecutando los comandos directamente en la ventana de comando, los resultados se obtienen inmediatamente en la ventana de resultados. Al trabajar de esta forma, la única manera de ir registrando todo lo realizado es mediante los archivos log. Sin embargo, esta forma de trabajar tiene la desventaja de que una vez que uno ha realizado varias modificaciones a la base de datos y uno quiere volver atrás, se pierde todo lo realizado y hay que volver a reconstruir todo con ayuda del log.

La manera más ordenada de trabajar en STATA cuando se requiere hacer varias modificaciones a la base de datos y obtener varias estadísticas de ella, es programar todos los comandos en un archivo do.

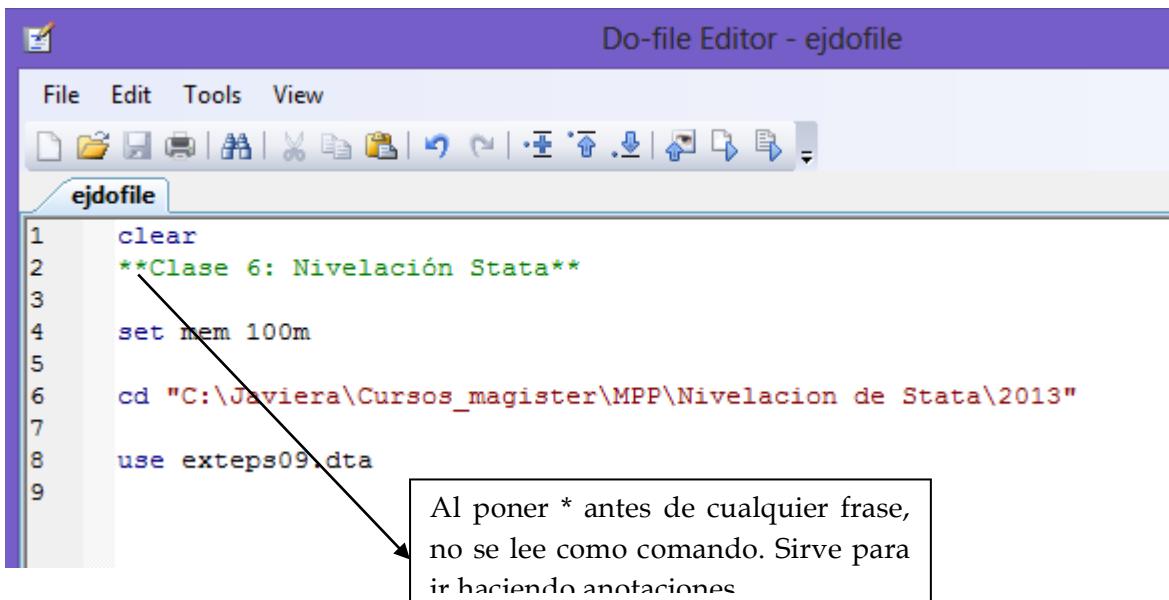
El archivo **do** no es mas que un archivo de texto que permite escribir las instrucciones para la ejecución de comandos en Stata.

Para abrir el archivo **do** debemos pinchar el icono  , y se abrirá la siguiente ventana:



Este icono es para correr el

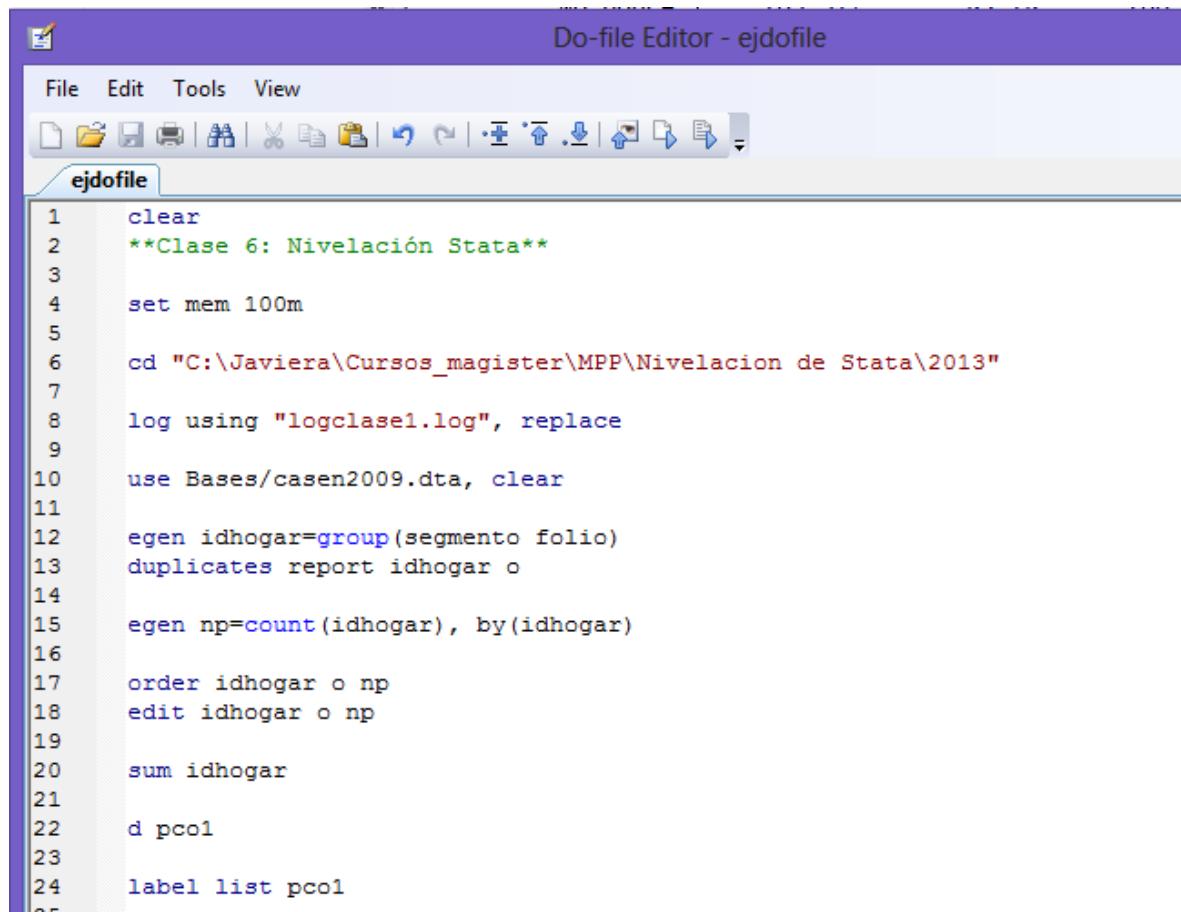
Por ejemplo, la forma típica de comenzar un **do** sería la siguiente:



Con esto ya he abierto la base de datos. A continuación puedo empezar a escribir los comandos para transformar la base de datos, para obtener estadísticas, etc. Exactamente de la misma forma que lo haría en la ventana de comandos pero ahora en forma más ordenada.

Es importante constantemente ir corriendo el do para detectar los errores que se están cometiendo.

En el cuadro siguiente observamos el archivo do file correspondiente al ejemplo visto en la sección 1;



The screenshot shows a Windows application window titled "Do-file Editor - ejdofile". The menu bar includes File, Edit, Tools, and View. Below the menu is a toolbar with various icons for file operations. The main area contains a code editor with the following Stata script:

```
1  clear
2  **Clase 6: Nivelación Stata**
3
4  set mem 100m
5
6  cd "C:\Javiera\Cursos_magister\MPP\Nivelacion de Stata\2013"
7
8  log using "logclase1.log", replace
9
10 use Bases/casen2009.dta, clear
11
12 egen idhogar=group(segmento folio)
13 duplicates report idhogar o
14
15 egen np=count(idhogar), by(idhogar)
16
17 order idhogar o np
18 edit idhogar o np
19
20 sum idhogar
21
22 d pco1
23
24 label list pco1
```

A continuación reportaremos las tasas de indigencia y pobreza para diferentes categorías:

a. Por sexo del jefe de hogar

En primer lugar será necesario crear una variable llamada sexojh, el cual tomará el valor de 1 si éste es hombre y 2 si éste es mujer. Luego se etiquetará la variable dándole el nombre de hombre cuando la variable sexojh tome el valor 1 y mujer cuando sexojh tome el valor 2.

```
. g sexojh=sexo if pco1==1  
(175464 missing values generated)  
. label define sexolbl 1 "Hombre" 2 "Mujer"  
. label values sexojh sexolbl
```

Ahora se generará una variable que tomará el valor de 1 para todos los miembros del hogar cuando esté es hombre y 2 para todos los miembros del hogar cuando este es mujer. Esto se hace generando una variable con el comando `egen` el cual genera una variable que toma el número mayor – asociado a la variable sexojh- por hogar; y como en los hogares donde el jefe de hogar es hombre el número mayor de la variable sexojh es 1 –lo otros son missings- y en los hogares donde el jefe es mujer ese número es 2.

```
. egen sexojhm=max(sexojh), by(idhogar)  
. edit idhogar o pco1 sexo sexojh sexojhm
```

Data Editor (Edit) - [casen2009]

File Edit Data Tools

var355[18]

	idhogar	o	pco1	sexo	sexojh	sexojhm
1	1	1	jefe(a)	hombre	Hombre	1
2	1	2	esposo(a)	mujer	.	1
3	1	3	nieto(a)	hombre	.	1
4	2	1	jefe(a)	hombre	Hombre	1
5	2	2	esposo(a)	mujer	.	1
6	2	3	hijo(a)	mujer	.	1
7	2	4	hijo(a)	mujer	.	1
8	3	1	jefe(a)	mujer	Mujer	2
9	3	2	nieto(a)	mujer	.	2
10	4	1	jefe(a)	hombre	Hombre	1
11	4	2	esposo(a)	mujer	.	1
12	4	3	hijo(a)	mujer	.	1
13	4	4	hijo(a)	mujer	.	1
14	4	5	hijo(a)	mujer	.	1

```
. drop sexojh
. rename sexojhm sexojh
. label values sexojh sexo
```

	o	pco1	sexo	idhogar	sexojh
1	1	jefe(a)	hombre	1	hombre
2	2	esposo(a)	mujer	1	hombre
3	3	nieto(a)	hombre	1	hombre
4	1	jefe(a)	hombre	2	hombre
5	2	esposo(a)	mujer	2	hombre
6	3	hijo(a)	mujer	2	hombre
7	4	hijo(a)	mujer	2	hombre
8	1	jefe(a)	mujer	3	mujer
9	2	nieto(a)	mujer	3	mujer

```
. tab pobre sexojh [w=expr], col nofreq  
(frequency weights assumed)
```

situacion de pobreza	sexojh		Total
	hombre	mujer	
Indigente	2.94	5.67	3.74
Pobre no indigente	10.01	14.67	11.38
No pobre	87.05	79.67	84.88
Total	100.00	100.00	100.00

Podemos decir que las personas que viven en hogares con jefes de hogar hombres tienen una tasa de indigencia es un 2.94%, y un porcentaje de pobres no indigentes igual a 10%, con lo cual la tasa de pobreza es un 12.9%. Por su parte las personas que viven en hogares con jefes de hogar mujer tienen una tasa de indigencia de 5.7%, y un porcentaje de pobres no indigentes un 14.7%, de esta forma la tasa de pobreza es 20.4%.

b. Número de personas del hogar

Se generará una variable llamada `categ_np` la cual divide a los hogares en 8 diferentes categorías, las cuales dependen del número de personas que tenga el hogar. Las categorías 1, son los hogares que tienen un solo miembro, la segunda categorías son los hogares que tienen dos miembros, y así hasta la octava categoría que incluye a los hogares que tienen 8 o más miembros.

```

. g categ_nps=nps
. replace categ_nps=8 if nps>8
(6034 real changes made)
. replace categ_nps=. if nps=.
(0 real changes made)
. tab categ_nps pobre [w=expr], row nofreq
(frequency weights assumed)

```

categ_nps	Situación de pobreza			Total
	Indigente	Pobre no	No pobre	
1	3.39	2.64	93.97	100.00
2	2.86	5.16	91.97	100.00
3	2.90	7.87	89.23	100.00
4	3.69	10.49	85.82	100.00
5	3.51	13.45	83.04	100.00
6	3.99	16.16	79.85	100.00
7	5.33	18.55	76.11	100.00
8	7.61	20.96	71.43	100.00
Total	3.74	11.38	84.88	100.00

En esta tabla se puede apreciar como las tasas de pobreza e indigencia aumentan a medida que aumenta el tamaño del hogar. En hogares de una persona el porcentaje de indigentes es 3.39%, y el porcentaje de pobres no indigentes es 2.64%. Por el contrario, en hogares con 8 personas o más, el porcentaje de indigentes de 7.61% y de pobres no indigentes 20.96.

c. Por educación del Jefe de hogar

En primer lugar será necesario crear una variable que determine el nivel de educación (años de educación) de las personas encuestadas y luego etiquetar aquellas variables. El nivel de educación viene determinado en la base de datos, sin embargo por ejercicio aquí se vuelve a generar una variable que determine el nivel educativo que la persona alcanzado:

```

g neduc=1 if esc==0
replace neduc=2 if esc>0 & esc<8
replace neduc=3 if esc==8
replace neduc=4 if esc>8 & esc<12

```

```

replace neduc=5 if esc==12
replace neduc=6 if esc>12
replace neduc=. if esc==.

label define neduclbl 1 "Sin educación" 2 "Básica Incompleta" 3 "Básica Completa"
4 "Media Incompleta" 5 "Media Completa" 6 "Superior"

label values neduc neduclbl

```

Luego de separar el nivel educativo de las personas en 6 niveles, se generará una variable que determinará el nivel de educación del jefe de hogar solamente y luego bajo el mismo proceso utilizado en la parte a se buscará el nivel de pobreza e indigencia por nivel educativo del jefe de hogar.

```

. g neducjh=neduc if pco1==1
(175464 missing values generated)
. egen neducjhm=max(neducjh), by(idhogar)
. drop neducjh
. rename neducjhm neducjh
. label values neducjh neduclbl

. tab neducjh pobre [w=expr], row norefq
(frequency weights assumed)

```

neducjh	Situación de pobreza			Total
	Indigente	Pobre no	No pobre	
Sin educación	5.06	16.82	78.12	100.00
Básica Incompleta	4.82	14.57	80.61	100.00
Básica Completa	5.95	16.77	77.28	100.00
Media Incompleta	4.80	13.95	81.25	100.00
Media Completa	2.76	10.26	86.99	100.00
Superior	1.45	3.01	95.54	100.00
Total	3.74	11.38	84.88	100.00

En la tabla anterior se puede apreciar claramente como la tasa de pobreza e indigencia disminuyen con el nivel educacional.

Capítulo VII: Aplicando lo aprendido II

En esta sección utilizaremos la encuesta de protección social del año 2009 (EPS2009), esta encuesta la pueden solicitar en el sitio web www.proteccionsocial.cl. Esta encuesta es del tipo longitudinal donde las mismas personas han sido entrevistadas en los años 2002, 2004, 2006 y 2009.

Para esta sección utilizaremos la base de historia laboral y la base entrevistado de esta encuesta.

Porcentaje de meses cotizados, trabajados y cesante

La base de historia laboral de la EPS2009 reporta cada uno de los estatus laborales de la persona entre Enero2006 y Mayo de 2009/Febrero de 2010, dependiendo de cuando fue entrevistada la persona.

En esta base de datos existe más de un registro (fila) por folio, donde cada observación representa el reporte de una condición laboral para un folio en particular. La siguiente figura muestra la estructura de dicha base de datos.

	folio	orden	b1im	b1ia	b2	bitm	b1ta	b4	oficio
1	179	1	enero	2006	inactivo	julio	2009	.	.
2	717	1	enero	2006	trabajan	junio	2009	15 regió	7111
3	785	1	enero	2006	inactivo	junio	2009	.	.
4	882	1	enero	2006	trabajan	junio	2009	2 región	9152
5	948	1	enero	2006	cesante	mayo	2009	.	.
6	1170	1	enero	2006	trabajan	septiemb	2009	15 regió	5230
7	1267	1	enero	2006	trabajan	mayo	2009	15 regió	5131
8	1401	1	enero	2006	trabajan	septiemb	2009	15 regió	5220
9	1563	1	enero	2006	trabajan	agosto	2009	15 regió	1314
10	1882	1	enero	2006	inactivo	junio	2007	.	.
11	1882	2	julio	2007	trabajan	octubre	2007	15 regió	4190
12	1882	3	noviembr	2007	inactivo	diciembr	2008	.	.
13	1882	4	enero	2009	trabajan	junio	2009	15 regió	4190
14	2016	1	enero	2006	trabajan	diciembr	2006	15 regió	1314
15	2016	2	enero	2007	inactivo	agosto	2009	.	.
16	2102	1	enero	2006	trabajan	junio	2009	1 región	9213
17	2240	1	enero	2006	inactivo	agosto	2009	.	.
18	2410	1	enero	2006	trabajan	septiemb	2009	15 regió	4115
19	2741	1	enero	2006	trabajan	julio	2007	4 región	9211
20	2741	2	agosto	2007	trabajan	agosto	2008	15 regió	5112
21	2741	3	septiemb	2008	trabajan	mayo	2009	2 región	9132
22	2741	4	junio	2009	cesante	junio	2009	.	.

Por ejemplo, la persona con folio 179 tiene una condición laboral reportada entre Enero de 2006 y Julio de 2009, que corresponde a inactividad. Por otra parte el folio 1882 tiene cuatro condiciones laborales reportadas entre Enero de 2006 y junio de 2009 (fecha en que se realizó la entrevista), la primera condición laboral (orden=1) indica que la persona estuvo inactiva entre Enero de 2006 y junio de 2007, la segunda (orden=2) indica que la persona estuvo trabajando entre Julio de 2007 y Octubre de 2007, la tercera (orden=3) indica que la persona estuvo inactiva entre Noviembre de 2007 y Diciembre de 2008, la cuarta (orden=4) indica que la persona estuvo trabajando entre Enero de 2009 y Junio de 2009 (momento de la entrevista).

Con esta información queremos construir para cada folio el número de meses total reportados en la historia laboral, lo que depende de cuando fue entrevistado, también la cantidad de meses trabajados en este periodo, la cantidad de meses cesante, y la cantidad de meses inactivo, así como dentro de los meses trabajados cuantos cotizó para el sistema de pensiones.

Definir la fecha de inicio y de término de cada condición laboral con el formato STATA:

```
use "historialaboral.dta", clear
```

```
g inicio=ym( b1ia, b1im)  
format inicio %tm
```

```
g termino=ym( b1ta, b1tm)  
format termino %tm
```

Generar una variable que contabilice el total de meses que dura cada condición laboral reportada

```
g total=termino-inicio+1
```

	folio	orden	b1im	b1ia	b2	b1tm	b1ta	total
1	179	1	enero	2006	inactivo	julio	2009	43
2	717	1	enero	2006	trabajan	junio	2009	42
3	785	1	enero	2006	inactivo	junio	2009	42
4	882	1	enero	2006	trabajan	junio	2009	42
5	948	1	enero	2006	cesante	mayo	2009	41
6	1170	1	enero	2006	trabajan	septiemb	2009	45
7	1267	1	enero	2006	trabajan	mayo	2009	41
8	1401	1	enero	2006	trabajan	septiemb	2009	45
9	1563	1	enero	2006	trabajan	agosto	2009	44
10	1882	1	enero	2006	inactivo	junio	2007	18
11	1882	2	julio	2007	trabajan	octubre	2007	4
12	1882	3	noviembr	2007	inactivo	diciembr	2008	14
13	1882	4	enero	2009	trabajan	junio	2009	6
14	2016	1	enero	2006	trabajan	diciembr	2006	12
15	2016	2	enero	2007	inactivo	agosto	2009	32
16	2102	1	enero	2006	trabajan	junio	2009	42
17	2240	1	enero	2006	inactivo	agosto	2009	44
18	2410	1	enero	2006	trabajan	septiemb	2009	45
19	2741	1	enero	2006	trabajan	julio	2007	19
20	2741	2	agosto	2007	trabajan	agosto	2008	13
21	2741	3	septiemb	2008	trabajan	mayo	2009	9
22	2741	4	junio	2009	cesante	junio	2009	1

Luego se generan los meses ocupado, meses cesante, meses inactivo y meses cotizados:

```
g mocupados=total if b2==1
```

```
replace mocupados=0 if mocupados==.
```

```
g mdesempleados=total if b2==2 | b2==3
```

```
replace mdesempleados=0 if mdesempleados==.
```

```
g minactivos=total if b2==4
```

```
replace minactivos=0 if minactivos==.
```

```
g mcotizados=total if b18>=1 & b18<=6  
replace mcotizados=0 if mcotizados==.
```

Luego se condensa la base de datos con el comando collapse, dejando sólo una observación por folio:

```
collapse (sum) total (sum) mocupados (sum) mdesempleados (sum) minactivos (sum)  
mcotizados, by(folio)
```

Luego pegamos esta información a la base de datos entrevistado (base principal de la Encuesta de Protección Social):

```
sort folio  
merge folio using "entrevistado.dta"
```

Una vez que a la base de datos principal de la EPS2006 le hemos pegado la información de meses ocupados, cotizados, desempleados e inactivos provenientes de la base de historia laboral podemos calcular algunas estadísticas de interés.

Densidad de cotización por género:

Primero generamos los meses cotizados sobre el total de duración de la historia laboral:

```
g decot= mcotizados/total*100
```

También podemos generar los meses cotizados sobre los meses trabajados:

```
g pcotocu= mcotizados/ mocupados*100
```

Y la proporción de la historia laboral que trabaja:

```
g pocu= mocupados/ total*100

. tabstat decot pocu pcotocu, stats(mean) by(a8)

a8 |      decot      pocu    pcotocu
-----+
hombre |  54.81203  74.90373  73.04477
mujer |  32.41956  44.78739  70.98873
-----+
Total |  43.37659  59.52385  72.19844
-----+
```

La densidad de cotización promedio entre hombres y mujeres difiere de manera importante, mientras los hombres cotizan en el sistema de pensiones un 54.8% del tiempo, las mujeres lo hacen un 32.4% del tiempo. Sin embargo, dichas diferencias tienen una explicación relativamente clara, mientras los hombres trabajan un 75% del tiempo, las mujeres sólo lo hacen un 44.8% del tiempo. Y dentro del tiempo trabajando, los hombres cotizan un 73% del tiempo y las mujeres un 71%, de esta forma lo que genera que la densidad de cotización entre hombres y mujeres difiera se debe principalmente a las diferencias en las tasas de ocupación.

Participación laboral por región

```
g pdesem= mdesempleados/ total*100
```

```
g pinact=minactivos/ total*100
```

```
tabstat pocu pdesem pinact, stats(mean) by(region)
```

Summary statistics: mean
by categories of: region (región de residencia)

region	pocu	pdesem	pinact
-----+-----			
1 región	58.36517	8.403273	33.23156
2 región	60.95686	5.314671	33.72847
3 región	61.68369	8.163529	30.15278
4 región	56.98716	11.79125	31.22159
5 región	57.12807	11.17482	31.69711
6 región	55.55487	10.14187	34.30325
7 región	56.81255	8.711744	34.47571
8 región	54.82597	8.838206	36.33582
9 región	55.38496	8.088677	36.52636
10 región	59.68335	7.008076	33.30857
11 región	63.30272	13.59173	23.10555
12 región	64.13892	7.954182	27.90689
r. metropolitana	63.82236	6.545224	29.63242
14 región (de lo	59.0257	3.211573	37.76272
15 región (arica	60.84679	6.301884	32.85133
-----+-----			
Total	59.52385	8.115695	32.36046
-----+-----			

Densidad de cotización por nivel educacional

```
gen teduc09=1 if a12n==2 | a12n==1 | a12n==5
```

```

replace teduc09=2 if (a12n==3) | (a12n==4 & a12c<8) | (a12n==6 & a12c==1) |
(a12n==8 & a12c==1)

replace teduc09=3 if (a12n==4 & a12c>=8 & a12c!=99) | (a12n==6 & a12c==2) |
(a12n==8 & a12c==2)

replace teduc09=4 if (a12n==6 & (a12c>=3 & a12c<=5)) | (a12n==7 & a12c<=3) |
(a12n==8 & (a12c>=3 & a12c<=5)) | (a12n==9 & a12c<=3)

replace teduc09=5 if (a12n==7 & a12c>=4 & a12c!=99) | (a12n==9 & a12c>=4 &
a12c!=99) | (a12n==6 & a12c>=6 & a12c!=99) | (a12n==8 & a12c>=6 & a12c!=99)

replace teduc09=6 if (a12n==10 | a12n==11)

replace teduc09=7 if (a12n==12 & a12c<5)

replace teduc09=8 if (a12n==12 & a12c>=5 & a12c!=99) | (a12n==13)

label define teduclbl 1 Ninguna 2 BasicaInc 3 BasicaCom 4 MediaInc 5 MediaCom 6
TecnicaSup 7 UnivInc 8 UnivCom

label values teduc09 teduclbl

label variable teduc09 "Nivel educacional EPS09"

```

```
tab teduc09 a8, summarize(decot) means
```

Means of decot

Nivel				
educaciona	a8. sexo			
1 EPS06	hombre	mujer	Total	
<hr/>				
Ninguna	15.211793	3.3484939	9.5380414	
BasicaInc	32.618014	10.52119	21.246106	
BasicaCom	53.61402	24.10269	39.208846	
MediaInc	60.969379	24.910989	43.571493	
MediaCom	69.558311	44.810543	56.549456	
TecnicaSu	72.705652	61.658043	66.526874	

```

UnivInc | 68.171117 57.278465 | 62.916036
UnivCom | 70.517907 71.743679 | 71.158401
-----+-----+-----
Total | 54.930629 32.529933 | 43.498226

```

Capítulo VIII: Crear matrices para guardar los datos

Una forma práctica de guardar la información proveniente de la aplicación del comando **sum** es a través de matrices. Cuando se hace un **sum** de una variable, en la memoria temporal del programa quedan guardadas las estadísticas; el nombre con el que cada una de éstas se guarda se puede saber al tipar **return list** después de hacer un sum.

```

. sum esc

      Variable |       Obs        Mean     Std. Dev.      Min      Max
-----+-----+-----+-----+-----+-----+
      esc |    180914     8.319069     4.286882          0         21

. return list

scalars:
      r(N) = 180914
      r(sum_w) = 180914
      r(mean) = 8.319068728788263
      r(Var) = 18.37735307412638
      r(sd) = 4.28688150922397
      r(min) = 0
      r(max) = 21
      r(sum) = 1505036

```

Lo mismo se puede hacer después de un **sum, detail**:

```
. sum esc, detail

                                escolaridad (años)
-----
Percentiles      Smallest
1%              0              0
5%              0              0
10%             2              0      Obs            180914
25%             5              0      Sum of Wgt.   180914

50%             8
                         Largest   Mean        8.319069
                         Std. Dev. 4.286882
75%            12             21
90%            13             21      Variance     18.37735
95%            15             21      Skewness    -.1286632
99%            17             21      Kurtosis    2.403091

. return list

scalars:
          r(N)      = 180914
          r(sum_w)  = 180914
          r(mean)   = 8.319068728788263
          r(Var)    = 18.37735307412638
          r(sd)     = 4.28688150922397
          r(skewness) = -.1286631597722507
          r(kurtosis) = 2.403091261120855
          r(sum)    = 1505036
          r(min)    = 0
          r(max)    = 21
          r(p1)     = 0
          r(p5)     = 0
          r(p10)    = 2
          r(p25)    = 5
          r(p50)    = 8
          r(p75)    = 12
          r(p90)    = 13
          r(p95)    = 15
          r(p99)    = 17
```

Supóngase que se requiere hacer la siguiente tabla de datos:

Genero	Ingreso promedio	Escolaridad promedio	Experiencia laboral promedio.
Hombre			
Mujer			

Esto puede hacerse generando una matriz de dos filas y tres columnas, para posteriormente completar sus elementos con las estadísticas correspondientes. Para esto primero se debe generar la matriz, llámémosla A.

```
matrix A=J(2,3,0)
```

Luego se hace el primer **sum**, de la variable ingreso para los hombres, y guardo este resultado en la posición [1,1] de la matriz:

```
sum ingreso if dhombre==1  
matrix A[1,1]=r(mean)
```

Y así sucesivamente hasta completar toda la matriz:

```
sum ingreso if dhombre==0  
matrix A[2,1]=r(mean)  
  
sum esc if dhombre==1  
matrix A[1,2]=r(mean)  
  
sum esc if dhombre==0  
matrix A[2,2]=r(mean)  
  
sum expr if dhombre==1  
matrix A[1,3]=r(mean)  
  
sum expr if dhombre==0  
matrix A[2,3]=r(mean)
```

Con esto se completa la matriz. Luego para que se vea, en Stata Results se tipea el comando **matriz list**:

```
. matrix list A
```

```
A[2,3]

      c1          c2          c3
r1  210198.54  8.3427264  58.336718
r2  162629.12  8.2959796  60.383309
```

Esta matriz se puede copiar, seleccionando la matriz en la ventana de resultados apretando el botón derecho y pinchando “copy table”; luego se puede llevar a un archivo Excel para su edición.

Capítulo IX: Ciclos recursivos

El comando **while** permite ejecutar una función en forma recursiva mientras cierta condición se cumpla. Por ejemplo:

```
local i

while `i'<22 {

tab dhombre if esc==`i'

local i=`i'+1

}
```

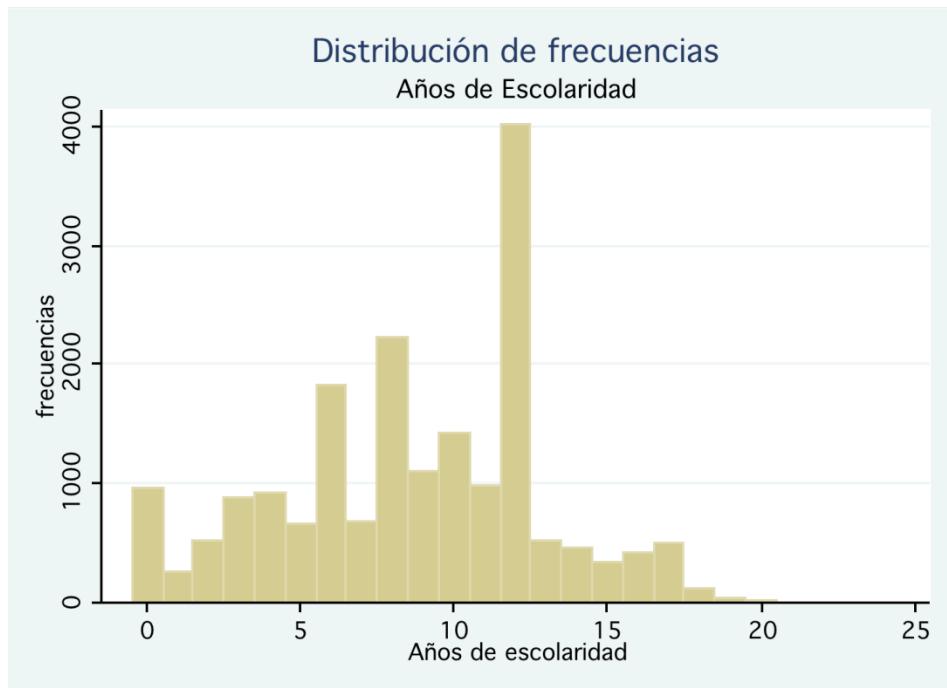
Este código hará 21 tablas, una para cada año de escolaridad, de la variable que identifica género.

Capítulo X: Análisis descriptivo de los datos

Distribución de frecuencias

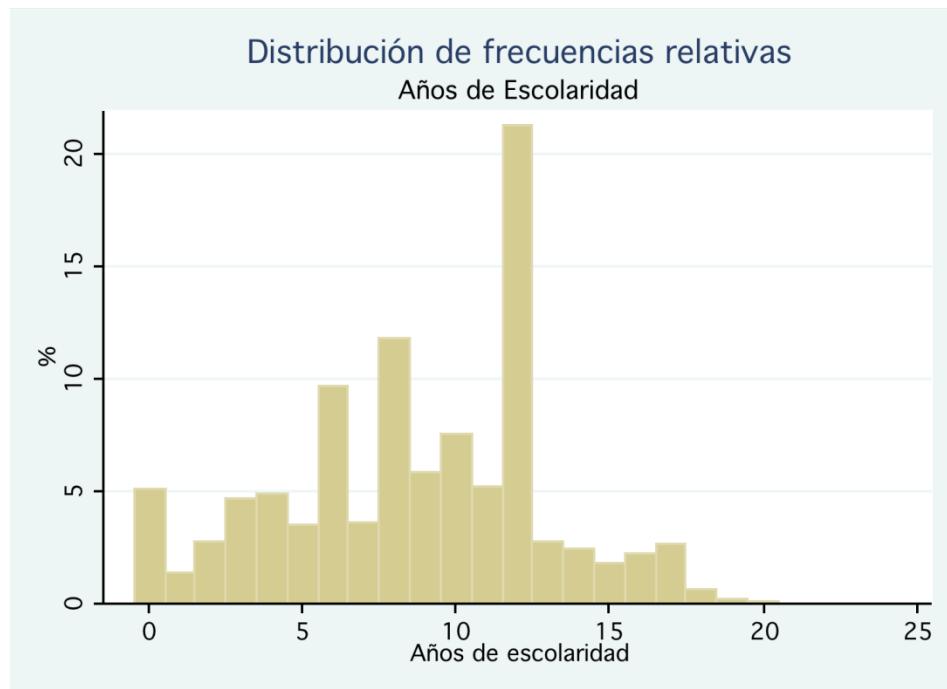
Cuando tenemos un conjunto de observaciones sobre una variable, por ejemplo años de escolaridad, existen varias formas de caracterizar la variable. Una forma de ilustrar tendencias y patrones de los datos de una variable dentro de la población es através de un gráfico de distribución de frecuencias. La distribución de frecuencias muestra en número de observaciones de la población que toma cada uno de los posibles valores de la variable. También podemos expresar la frecuencia de cada valor de la variable como una fracción o porcentaje lo que se conoce como frecuencia relativa. Un histograma nos muestra una distribución de frecuencias, por ejemplo, el siguiente gráfico corresponde a la distribución de frecuencias de los años de escolaridad de una población de 18.989 individuos:

```
histogram esc, discrete frequency ytitle(frecuencias) xtitle(Años de escolaridad) title(Distribución de frecuencias) subtitle(Años de Escolaridad)
```



Y el siguiente gráfico nos muestra la distribución de frecuencias relativas:

```
histogram esc, discrete percent ytitle(frecuencias) xtitle(Años de  
escolaridad) title(Distribución de frecuencias) subtitle(Años de  
Escolaridad)
```



La distribución de frecuencias también se puede ver mediante una tabla con los datos:

```
. tab esc
```

escolaridad	Freq.	Percent	Cum.
0	960	5.08	5.08
1	254	1.34	6.42
2	518	2.74	9.16
3	892	4.72	13.89
4	923	4.88	18.77
5	664	3.51	22.28

6	1,833	9.70	31.98
7	674	3.57	35.55
8	2,234	11.82	47.37
9	1,104	5.84	53.21
10	1,432	7.58	60.79
11	992	5.25	66.04
12	4,012	21.23	87.27
13	517	2.74	90.00
14	462	2.44	92.45
15	338	1.79	94.24
16	413	2.19	96.42
17	500	2.65	99.07
18	116	0.61	99.68
19	45	0.24	99.92
20	13	0.07	99.99
22	1	0.01	99.99
23	1	0.01	100.00
-----+-----			
Total	18,898	100.00	

Estadísticas descriptivas

El histograma o distribución de frecuencias sólo nos permite ilustrar en forma general el patrón de la variable, sin embargo, la mayor parte del tiempo se necesitan medidas mas exactas como las estadísticas descriptivas, dos características fundamentales de una variable son las medidas de tendencia central y medidas de dispersión.

Tendencia central: se refiere al punto medio de una distribución. Las medidas de tendencia central se conocen como medidas de posición.

```
set obs 3000

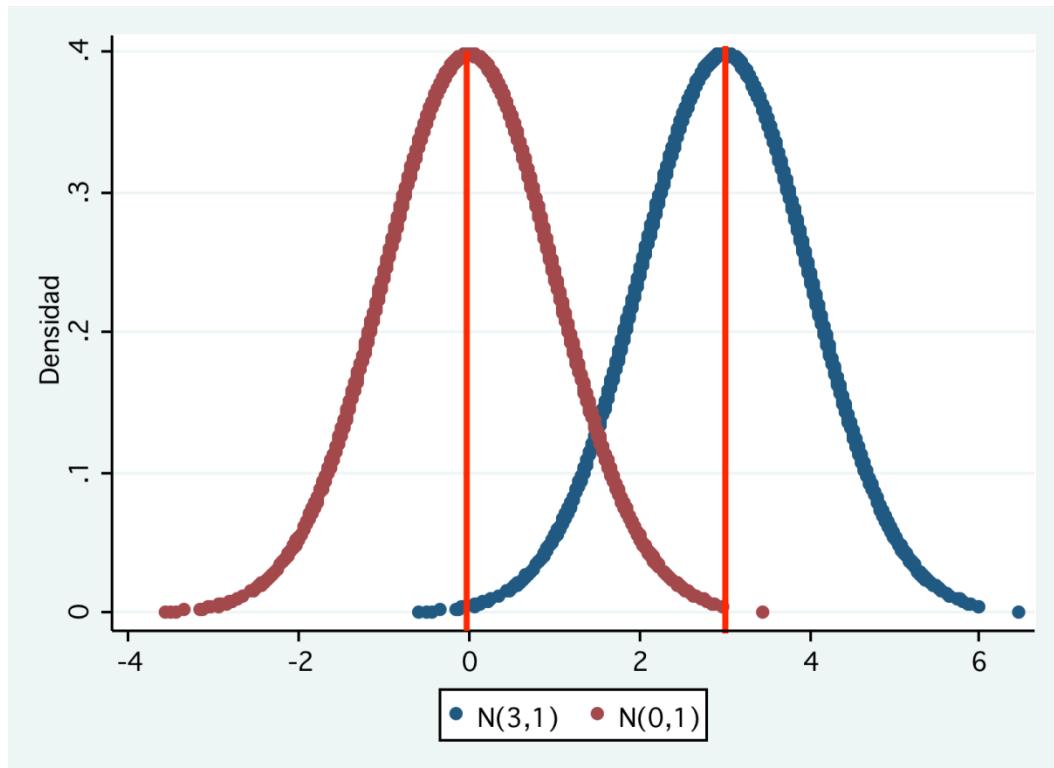
g z=uniform()

g normal_01=invnormal(z)

g dennormal_01=normalden(normal_01)

g normal_31=normal_01+3
```

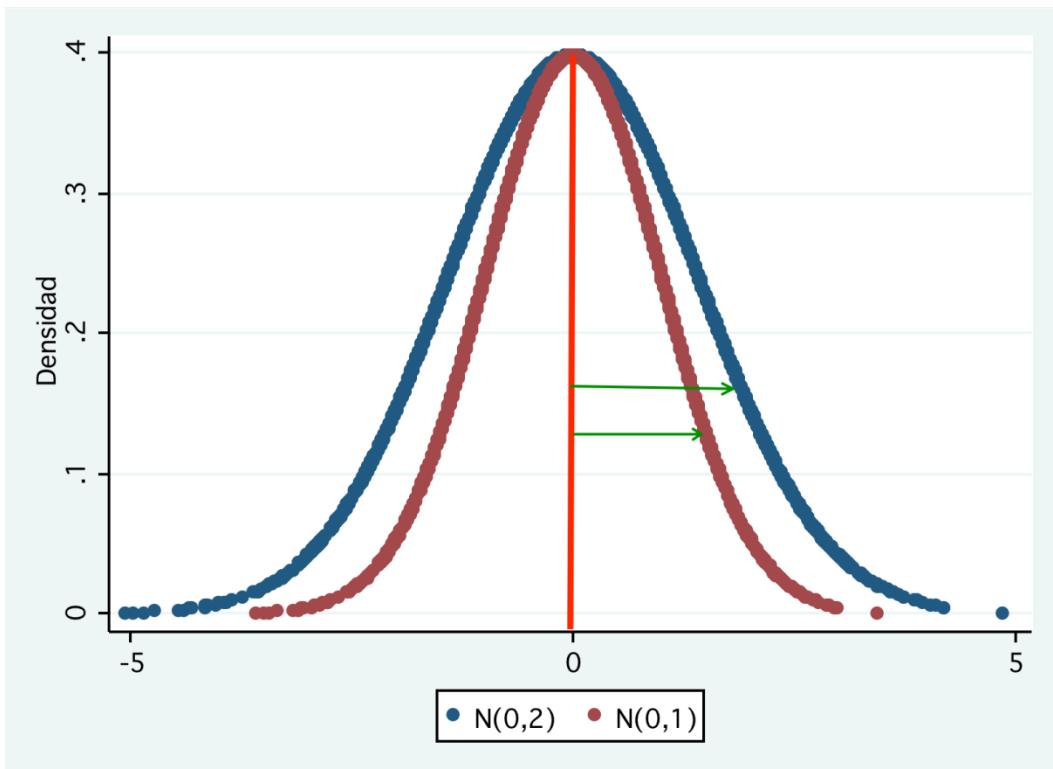
```
twoway (scatter dennormal_01 normal_31) (scatter dennormal_01
normal_01), ytitle(Densidad) legend(order(1 "N(3,1)" 2 "N(0,1)"))
```



Dispersión: se refiere a la separación de los datos de una distribución, es decir, al grado en que las observaciones se separan.

```
g normal_02=normal_01*sqrt(2)
```

```
twoway (scatter dennormal_01 normal_02) (scatter dennormal_01
normal_01), ytitle(Densidad) legend(order(1 "N(0,2)" 2 "N(0,1)"))
```



Sesgo: las curvas que representan los datos pueden ser simétricas o sesgadas. Cuando la curva es simétrica la media divide a la distribución en dos partes iguales. El siguiente gráfico nos muestra la comparación de una distribución simétrica como la normal, con una distribución asimétrica como la distribución Pareto⁴:

```
set obs 3000
g u=uniform()
scalar b=0.2
scalar a=5
g x=b/((1-u)^(1/a))
```

⁴ Una variable x que se distribuye *pareto* tiene la siguiente función de distribución de densidad y probabilidad acumulada:

$$f(x) = \frac{\alpha\beta^\alpha}{x^{\alpha+1}} \quad \alpha > 1; \beta > 0; x > \beta$$

$$F(x) = 1 - \left(\frac{b}{x}\right)^\alpha$$

```

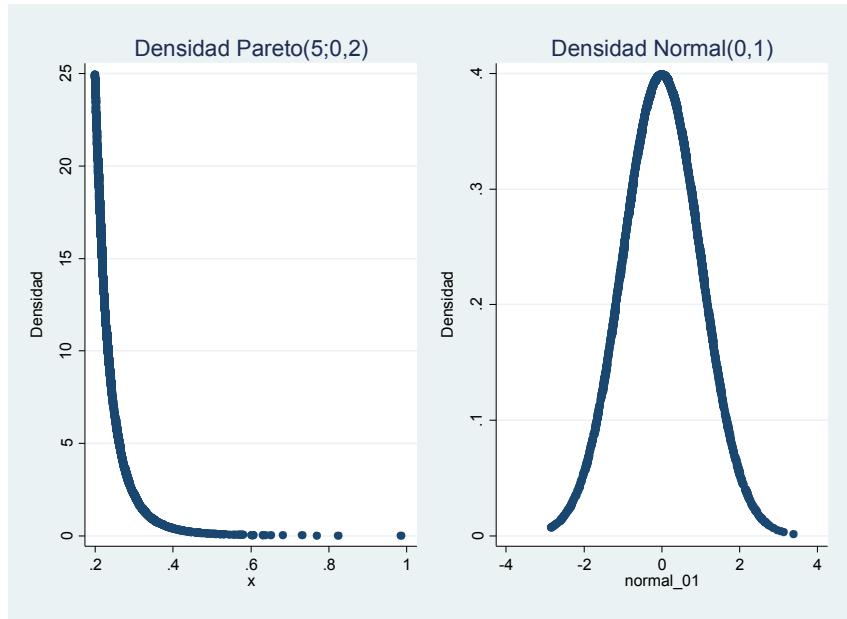
g pareto=a*(b^a) / (x^(a+1))

twoway (scatter pareto x), ytitle(Densidad) title(Densidad Pareto(5;0,2))
name(pareto, replace)

twoway (scatter dennormal_01 normal_01), ytitle(Densidad) title(Densidad
Normal(0,1)) name(normal)

graph combine pareto normal

```



Curtosis: la curtosis de la distribución nos dice que tan puntiaguda es la distribución

Medidas de tendencia central

Al describir grupos de observaciones, con frecuencia se desea describir el grupo con un solo número. Para tal fin, desde luego, no se usará el valor más elevado ni el valor más pequeño como único representante, ya que solo representan los extremos más bien que valores típicos. Entonces sería más adecuado buscar un valor central. Las medidas que describen un valor típico en un grupo de observaciones suelen llamarse medidas de tendencia central.

Media aritmética (media o promedio): corresponde a la medida de tendencia central más utilizada, corresponde a la suma de todas las observaciones divididas por la cantidad de observaciones. Esta medida es un buen indicador de la variable cuando la distribución de la misma es relativamente simétrica, no así cuando existen valores extremos (outliers) o la distribución es muy asimétrica.

Por ejemplo, en la CASEN 2006 viene la variable ytotaj que corresponde a los ingresos totales individuales. El siguiente comando nos muestra el número de observaciones de esta variable y su promedio:

```
. sum ytotaj
```

Variable	Obs	Mean	Std. Dev.	Min	Max
ytotaj	152132	213013.5	514864.6	42	6.39e+07

El promedio del ingreso total individual es de \$213,013.5, sin embargo, un 73.5% de estas observaciones tienen un ingreso inferior al promedio, por lo tanto, cuando la variable tiene una distribución muy asimétrica como es el caso del ingreso, por la presencia de valores extremos, el promedio deja de ser una buena medida de tendencia central.

```
. count if ytotaj < r(mean) & ytotaj != .
```

```
111787
```

```
. di 111787/152132
```

```
.73480267
```

Por lo tanto, SOLO cuando la distribución de la variable es relativamente simétrica la media o promedio es una buena medida de tendencia central.

Moda: corresponde al valor que más se repite en las observaciones, esta medida tiene más sentido cuando la variable es discreta, ya que cuando la

variable es continua con poca probabilidad se repetirán valores y si se repiten no necesariamente representa una medida de tendencia central.

Percentiles: representa el valor de la variable para el individuo u observación ubicado en cierto porcentaje de la población, por ejemplo, la **mediana** o percentil 50 representa el valor de la variable de la observación ubicada justo en la mitad de las observaciones, si una variable tiene 5 observaciones y las ordenamos de menor a mayor, la mediana de esta variable es el valor de "la persona 3", si son 6 observaciones la media es el promedio de los valores de las personas 3 y 4. De forma análoga, el percentil 10 representa el valor de la variable que tiene la persona que está justo en el 10% de menores valores de la variable, y así sucesivamente.

La siguiente tabla nos muestra los percentiles del ingreso total:

ingreso total				
	Percentiles	Smallest		
1%	4126	42		
5%	4126	42		
10%	4126	42	Obs	152132
25%	47186	63	Sum of Wgt.	152132
50%	126250		Mean	213013.5
		Largest	Std. Dev.	514864.6
75%	225200	3.07e+07		
90%	435203	3.53e+07	Variance	2.65e+11
95%	660748	3.63e+07	Skewness	31.37304
99%	1636128	6.39e+07	Kurtosis	2349.794

En muchos análisis interesa dividir la población en grupos según quintiles o deciles de alguna variable, por ejemplo, cuando sea analiza la distribución de ingresos nos interesa dividir a la población según quintiles de ingreso. Lo que usualmente se hace es tomar el ingreso total autónomo del hogar y dividirlo por el número de integrantes del mismo, donde se excluye el servicio doméstico. Luego se divide a la población en cinco (quintiles) o 10 (deciles) partes iguales según esta variable, y luego se pueda analizar distintas

características de estos quintiles/deciles. Vemos como se realiza esto en STATA con la CASEN 2006:

```
use "C:\Javiera\Casen\Casen_2006\casen2006\casen2006.dta", clear
egen hogar=group(r p comuna seg z f)

g uno=1
replace uno=0 if pcol==14
egen np=sum(uno), by(hogar)
g yaupc= yauthaj /np

xtile quintil=yaup if pcol==1 [w=expr], nq(5)

tab quintil [w=expr]

5 quantiles |
    of yaup |      Freq.      Percent      Cum.
-----+-----
      1 |  813,736      20.00      20.00
      2 |  813,835      20.00      40.00
      3 |  813,944      20.01      60.01
      4 |  813,597      20.00      80.01
      5 |  813,371      19.99     100.00
-----+-----
Total |  4,068,483      100.00
```

Con este comando solo se ha creado la variable quintil para los jefes de hogares, si queremos que quede plasmada en cada uno de los integrantes debemos realizar la siguiente operación:

```
egen quintil_2=max(quintil), by(hogar)
drop quintil
```

```
rename quintil_2 quintil  
replace quintil=. if pcol==14
```

Medidas de dispersión

Las medidas de dispersión muestran la variabilidad de una variable o de su distribución, indicando por medio de un número cuan alejadas están las observaciones de la media.

Varianza/Desviación estándar: la **varianza** mide el promedio de las diferencias al cuadrado de la observación con respecto a la media. Esta medida no tiene una interpretación fácil, pero si la tiene la **desviación estándar** que corresponde a la raíz cuadrada de la varianza. La desviación estándar queda expresada en las mismas unidades de la variable, por lo cual tiene una interpretación más directa.

```
. tabstat yaupc esc, stats(sd) by(quintil)
```

Summary statistics: sd

by categories of: quintil

quintil	yaupc	esc
1 14464.72	4.00379	
2 10474.74	3.968479	
3 14075.62	4.086183	
4 31462.44	4.120101	
5 621575.5	4.282641	
Total 250734.9	4.318203	

Rango: corresponde a la diferencia entre el valor mínimo y el máximo de las observaciones. Cuando existen valores muy extremos (distribución muy asimétrica) una mejor medida de dispersión es el **rango intercuartil** que corresponde a la diferencia entre el percentil 75 y el percentil 25.

Coeficiente de variación: como ya se mencionaba antes ya desviación estándar es una medida de dispersión que queda expresada en las mismas unidades de la variable, por lo cual, no es posible comparar la desviación estándar de dos variables medidas en distintas unidades, por ejemplo, peso y altura. Entonces cuando nos interesa comparar la dispersión de dos variables, se utiliza el **coeficiente de variación**, indica la desviación estándar como proporción de la media de la variable.

```
. tabstat yaupc esc [w=expr], stats(cv iqr) by(quintil)
(analytic weights assumed)

Summary statistics: cv, iqr

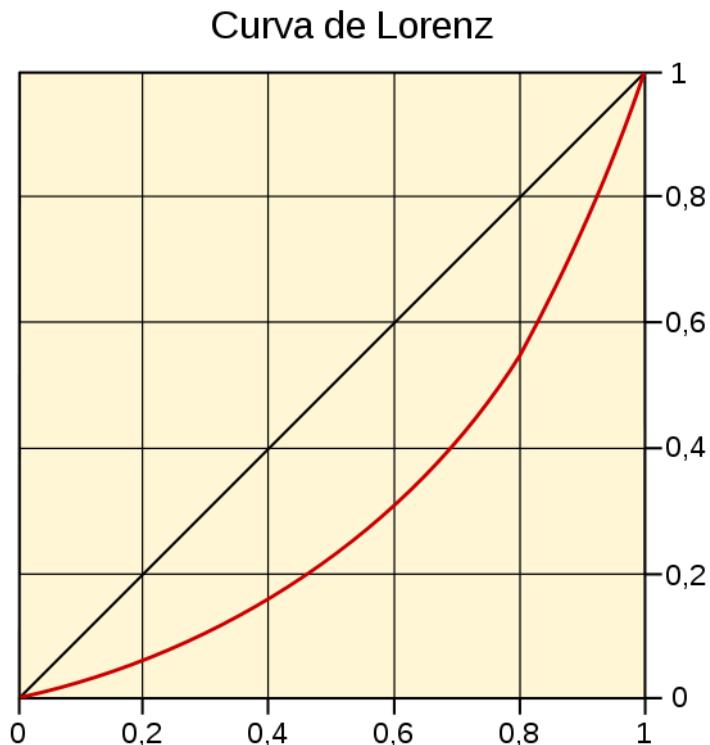
by categories of: quintil

quintil |      yaupc        esc
-----+-----
    1 |    .448922    .4830126
          | 20995.53         6
-----+-----
    2 |    .1533351    .4273353
          |   17904         6
-----+-----
    3 |    .1288017    .4002844
          | 24030.31         4
-----+-----
    4 |    .171898    .3501379
          | 52202.25         4
-----+-----
    5 |    .9643237    .2917928
          | 299214.7          4
-----+-----
  Total |  1.671222    .4160127
          | 127589.5          4
-----+
```

Medidas de desigualdad

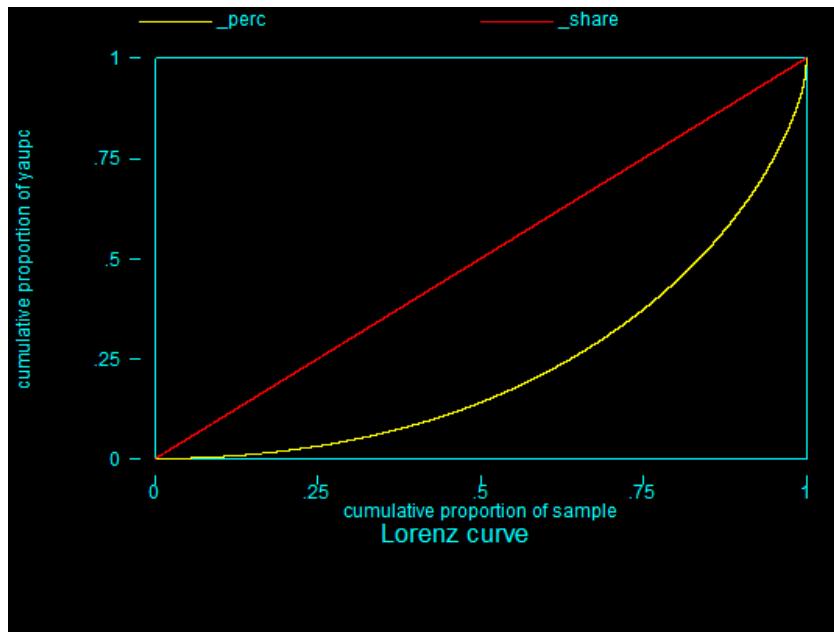
En esta sección sólo veremos dos de las medidas de desigualdad más conocidas y utilizadas en la literatura de distribución del ingreso, la Curva de Lorenz, y el coeficiente de GINI. Otras medidas de desigualdad utilizadas son la razón de percentiles, que ya vimos en la sección anterior como calcular.

Curva de Lorenz: representa el porcentaje acumulado del ingreso total en manos del porcentaje acumulado de la población.



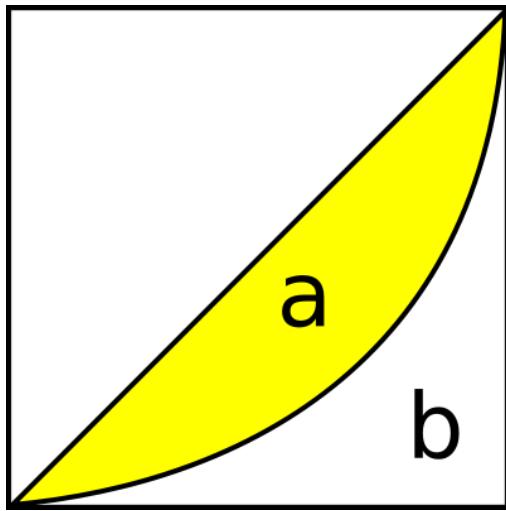
En STATA el comando `lorenz` nos grafica la Curva de Lorenz para la variable que indiquemos, por ejemplo el siguiente gráfico nos muestra la Curva de Lorenz del ingreso autónomo per cápita:

lorenz yaupc



Coeficiente de GINI: es una medida de la desigualdad que normalmente se utiliza para medir la desigualdad en los ingresos, pero puede utilizarse para medir cualquier forma de distribución desigual. El coeficiente de Gini es un número entre 0 y 1, en donde 0 se corresponde con la perfecta igualdad (todos tienen los mismos ingresos) y 1 se corresponde con la perfecta desigualdad (una persona tiene todos los ingresos y todos los demás ninguno). El **índice de Gini** es el coeficiente de Gini expresado en porcentaje, y es igual al coeficiente de Gini multiplicado por 100.

El coeficiente de Gini se calcula como una ratio de las áreas en el diagrama de la curva de Lorenz. Si el área entre la línea de perfecta igualdad y la curva de Lorenz es A, y el área por debajo de la curva de Lorenz es B, entonces el coeficiente de Gini es $A/(A+B)$. Esta ratio se expresa como porcentaje o como equivalente numérico de ese porcentaje, que es siempre un número entre 0 y 1.



El comando `inequal` de STATA nos entrega, dentro de otros indicadores, el coeficiente de GINI:

```
. inequal yaupc
inequality measures of yaupc
-----
relative mean deviation           .38951663
coefficient of variation          1.9316582
standard deviation of logs        .9746129
Gini coefficient                .53560216
Mehran measure                   .66121739
Piesch measure                   .47279453
Kakwani measure                  .23941463
Theil entropy measure            .6242073
Theil mean log deviation measure .48577899
```

La interpretación del coeficiente de GINI es la siguiente, si yo tengo tomo dos familias o personas aleatoriamente, la diferencia de ingresos de estas dos personas como proporción del ingreso promedio:

$$\Delta y = \frac{y_i - y_j}{\bar{y}}$$

Es el doble del coeficiente de GINI. En nuestro ejemplo, si yo tomo dos personas aleatoriamente, y considerando que el ingreso autónomo per-cápita

promedio es de \$134.061, la diferencia en ingresos de estas dos personas es 107,12% este ingreso medio.

Test de Hipótesis sobre la media poblacional

La idea detrás de hacer una prueba de hipótesis es que como investigador tenemos una hipótesis sobre un parámetro poblacional, por ejemplo la media. Para estimar este parámetro poblacional generalmente recurrimos a una muestra de observaciones de la población, luego utilizamos el test de hipótesis para decir que tan probable es que nuestro parámetro poblacional hipotético sea correcto. Para esto tomamos el promedio muestral y el valor poblacional supuesto, y vemos si esta diferencia es significativa o no.

En una prueba de hipótesis debemos establecer el valor supuesto o hipotético del parámetro poblacional antes de tomar la muestra. La suposición que deseamos probar se conoce como hipótesis nula y se simboliza como H_0 . Supongamos que queremos testear que el ingreso autónomo promedio de la población chilena es de 300 mil pesos:

$$H_0: \mu = \mu_{H_0} = 300,000$$

Si los resultados de la muestra no respaldan que se cumple la hipótesis nula debemos concluir que se cumple otra cosa. Siempre que rechazamos la hipótesis nula, la conclusión que si aceptamos se llama hipótesis alternativa (H_1). Para la hipótesis nula anterior se pueden considerar tres hipótesis alternativas:

$$H_1: \mu \neq 300,000$$

$$H_1: \mu > 300,000$$

$$H_1: \mu < 300,000$$

La siguiente tabla nos muestra el promedio y desviación estándar obtenido de la muestra de datos correspondientes a la encuesta CASEN 2006:

```
. sum yautaj

Variable |       Obs        Mean    Std. Dev.      Min      Max
-----+-----+-----+-----+-----+-----+
yautaj | 121895  259172.3   565043.2      42  6.39e+07
```

Con estos datos podemos construir el estadístico que nos permite testear la hipótesis nula:

$$t = \frac{\bar{x} - \mu}{\sqrt{V(\bar{x})}} \sim t_{n-1}$$

Donde:

$$V(\bar{x}) = \frac{V(x)}{n}$$

Y por lo tanto, el estadístico t es:

$$t = \sqrt{n} \frac{\bar{x} - \mu}{\sqrt{V(\bar{x})}} \sim t_{n-1}$$

Reemplazando los valores correspondientes:

$$t = \sqrt{121895} \cdot \left(\frac{326896.7 - 300000}{613375.2} \right)$$

```
. di sqrt(r(N)) * (r(mean)-300000) / r(sd)
-25.227046
```

Para ver la probabilidad asociada a este estadístico:

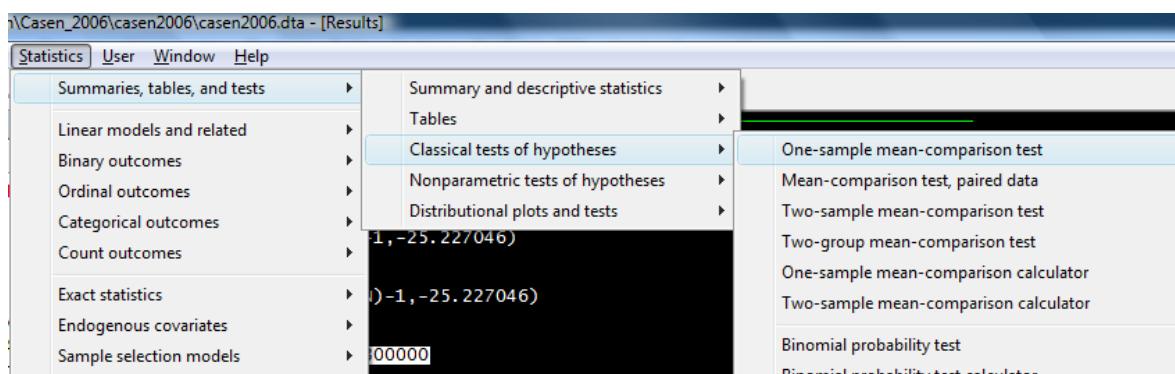
```
. di 1-ttail(r(N)-1, -25.227046)
0
```

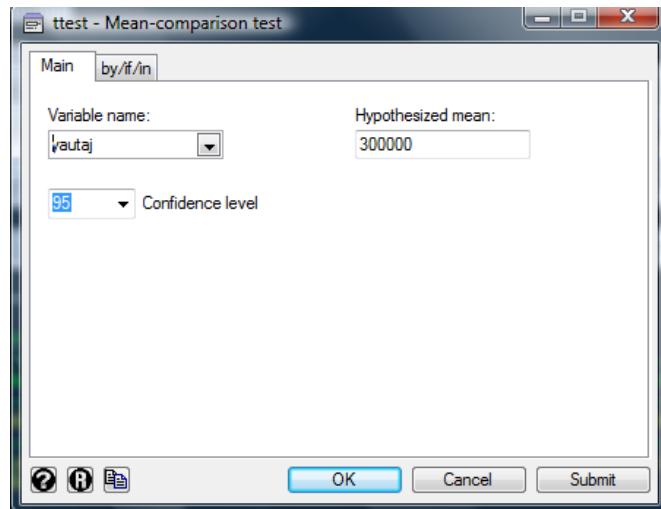
Este mismo test de hipótesis se puede realizar en un solo paso con el siguiente comando en STATA:

```
. ttest yautaj=300000

One-sample t test
-----
Variable |   Obs        Mean    Std. Err.    Std. Dev. [95% Conf. Interval]
-----+
yautaj | 121895    259172.3     1618.411    565043.2    256000.2    262344.3
-----
mean = mean(yautaj)                                     t = -25.2270
Ho: mean = 300000                                     degrees of freedom = 121894
Ha: mean < 300000          Ha: mean != 300000          Ha: mean > 300000
Pr(|T| < t) = 0.0000          Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

También podemos utilizar las ventanas de STATA para realizar el mismo test:





Obteniendo exactamente el mismo resultado.

Para concluir el resultado sobre la hipótesis nula debemos escoger un nivel de significancia, este usualmente es de 5% y representa el error que se esta dispuesto a tolerar en la interpretación de los resultados. El valor del estadístico t con 5% de significancia y $n-1$ grados de libertad se puede encontrar en la tabla de la distribución t-student, para mas de 120 grados de libertad este valor es de 1.98. Si el estadístico calculado es mayor a 1.96 o menor a -1.96 podemos rechazar la hipótesis nula con un 95% de seguridad o de confianza. En este caso el estadístico calculado es claramente menor al valor de tabla (-1.96) rechazando la hipótesis nula de que la media del ingreso autónomo es de 300 mil pesos. Una forma alternativa de concluir sobre la hipótesis nula una vez establecido el nivel de significancia es viendo el p-value, este valor es el nivel de significancia asociado al estadístico calculados, si este es menor al nivel de significancia escogido inicialmente, se rechaza la hipótesis nula. La tercera forma de testear a hipótesis nula es con el intervalo de confianza, si hemos escogido un 5% de significancia, el intervalo de confianza nos dice el rango de valores más probable (con 95% de seguridad) de la media, en este caso, con 95% de seguridad la media esta entre \$256,000.2 y \$262,344.3, como \$300,000 esta fuera de los valores más probable se rechaza la hipótesis nula de que la media sea igual a 300 mil.

También podemos escoger un nivel de significancia de 1%:

```
. ttest yautaj == 300000, level(99)

One-sample t test
-----
Variable |   Obs        Mean    Std. Err.    Std. Dev. [99% Conf. Interval]
-----+
yautaj | 121895    259172.3     1618.411    565043.2    255003.5    263341.1
-----
mean = mean(yautaj)                                     t = -25.2270
Ho: mean = 300000                                     degrees of freedom = 121894
Ha: mean < 300000          Ha: mean != 300000          Ha: mean > 300000
Pr(T < t) = 0.0000          Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

No existe un nivel de significancia establecido, sin embargo, es común que se utilice el 5% de significancia en la mayoría de los casos. Debemos tener presente que mientras mayor es el nivel de significancia escogido mayor será la probabilidad de rechazar la hipótesis nula cuando esta es cierta (cometer error Tipo I).

Test de diferencia de medias

De igual forma, podemos realizar test de diferencia de medias entre dos variables, o de la misma variable pero medias de distintos grupos.

Diferencia de medias de dos variables:

Este test se puede realizar de dos maneras, testeando que en promedio la diferencia entre las dos variables es igual a cero (paired), o testeando que la diferencia de los promedios de los dos variables es igual a cero (unpaired).

Por ejemplo, podríamos testear que en promedio la diferencia entre el ingreso autónomo individual y ingreso autónomo per-cápita es igual a cero:

```
. ttest yautaj == yaupc

Paired t test
-----  

Variable |   Obs      Mean   Std. Err.   Std. Dev. [95% Conf. Interval]  

-----+-----  

yautaj | 121892  259175.3    1618.449   565049.8  256003.2  262347.5  

yaupc | 121892  164064.2    860.9343   300578.3  162376.8  165751.6  

-----+-----  

diff | 121892  95111.11    1210.12    422489.7  92739.29  97482.92  

-----  

mean(diff) = mean(yautaj - yaupc)          t = 78.5964  

Ho: mean(diff) = 0                         degrees of freedom = 121891  

Ha: mean(diff) < 0             Ha: mean(diff) != 0           Ha: mean(diff) > 0  

Pr(T < t) = 1.0000            Pr(|T| > |t|) = 0.0000        Pr(T > t) = 0.0000
```

En este caso se rechaza la hipótesis nula de en promedio la diferencia de estas dos variables es igual a cero.

La otra forma, es testear que la diferencia de los promedios de las dos variables es igual a cero:

```
. ttest yautaj == yaupc, unpaired

Two-sample t test with equal variances
-----  

Variable |   Obs      Mean   Std. Err.   Std. Dev. [95% Conf. Interval]  

-----+-----  

yautaj | 121895  259172.3    1618.411   565043.2  256000.2  262344.3  

yaupc | 252066  134060.9    515.7929   258959.9  133050   135071.9  

-----+-----  

combined | 373961  174841.8    639.0276   390780.4  173589.3  176094.3  

-----+-----  

diff |          125111.4    1347.877          122469.6  127753.2  

-----  

diff = mean(yautaj) - mean(yaupc)          t = 92.8210  

Ho: diff = 0                         degrees of freedom = 373959  

Ha: diff < 0             Ha: diff != 0           Ha: diff > 0  

Pr(T < t) = 1.0000            Pr(|T| > |t|) = 0.0000        Pr(T > t) = 0.0000
```

Donde también se rechaza la hipótesis nula de que la diferencia de los promedios de estas variables sea igual a cero.

Diferencia de media entre grupos

Existen otros casos en que nos interesa testear sobre una misma variable la diferencia de medias entre grupos, por ejemplo, testear si la diferencia entre el ingreso promedio autónomo de hombres y mujeres es estadísticamente significativa:

```
. ttest yautaj, by(sexo)

Two-sample t test with equal variances
-----+
      Group |      Obs        Mean    Std. Err.    Std. Dev.   [95% Conf. Interval]
      +-----+
        hombre |    74524    300655.7     2276.06    621343.6    296194.6    305116.8
        mujer  |    47371    193910.6     2091.574    455228.5    189811.1    198010.1
      +-----+
      combined |   121895    259172.3     1618.411    565043.2    256000.2    262344.3
      +-----+
      diff    |          106745.1     3306.152
      +-----+
      diff = mean(hombre) - mean(mujer)                      t = 32.2868
Ho: diff = 0                                              degrees of freedom = 121893
      Ha: diff < 0          Pr(T < t) = 1.0000
      Ha: diff != 0         Pr(|T| > |t|) = 0.0000
      Ha: diff > 0          Pr(T > t) = 0.0000
```

Vemos que la diferencia entre el promedio de ingresos autónomos de hombres y mujeres es aproximadamente 106 mil pesos, la cual es estadísticamente significativa.

Covarianza y Correlación

Cuando estamos interesados no sólo en una variable, sino en como se relaciona esta con otra variable podemos utilizar la covarianza y/o correlación.

La covarianza muestra en valor esperado como se relacionan las dos variables, la dificultad es que este indicador no tienen una escala de interpretación estándar. Por eso es conveniente utilizar la correlación que la covarianza pero estandarizada, el coeficiente de correlación siempre estará entre -1 y 1, donde -1 indica una alta correlación negativa entre las variables y 1 indica una alta

correlación positiva entre las variables, si la correlación es cero indica que no existe una relación entre las variables.

Las siguientes tablas muestran como computar las covarianzas y correlaciones en STATA:

```
. correlate esc yautaj, covariance  
(obs=119432)
```

	esc	yautaj
esc	19.6016	
yautaj	564191	3.2e+11

Varianzas

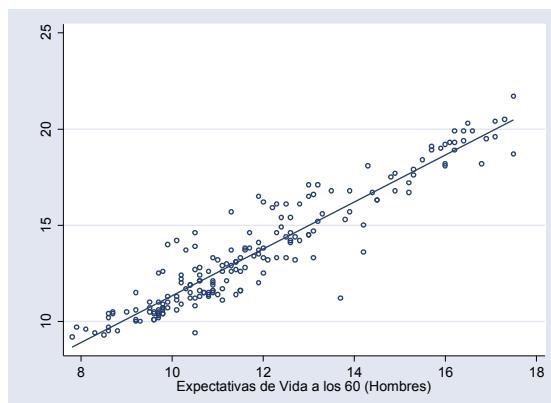
```
. correlate esc yautaj  
(obs=119432)
```

	esc	yautaj
esc	1.0000	
yautaj	0.2237	1.0000

Coeficiente
de

También podemos mediante inspección gráfica ver el grado de asociación entre dos variables, este tipo de gráfico se denomina diagrama de dispersión (scatterplot). El gráfico 1 muestra una asociación positiva entre las expectativas de vida de los hombres y de las mujeres a los 60 años para un conjunto de 188 países según datos de la Organización Mundial de la Salud. Como las observaciones están bastante cercanas a la recta de regresión, se puede decir que la correlación exhibida es fuerte. De hecho, el coeficiente de correlación es 0,94. Se puede apreciar, además, que la pendiente de la recta es mayor que 1. Esto ocurre porque en casi todos los países, las mujeres tienen expectativas de vida mayores que las de los hombres.

Gráfico 1



Nota:

- Primero se llaman los datos:

```
use who2001, clear
```

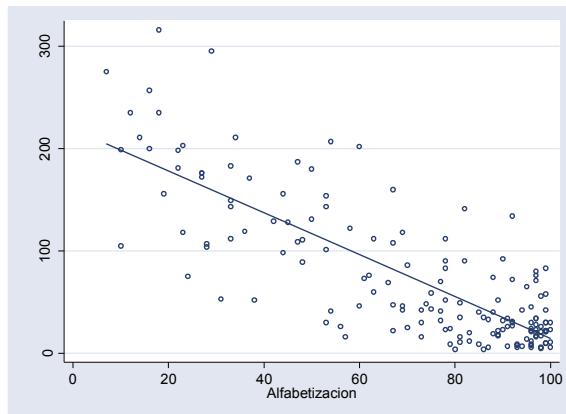
- Luego se usa el comando “graph twoway” con la siguiente instrucción:

```
graph twoway (lfit lex60_f lex60_m)(scatter lex60_f lex60_m, mstyle(p1) ms(oh) ),  
xtitle("Expectativas de Vida a los 60 (Hombres)") ytitle("Exp. vida 60 (Mujeres)")  
legend(off) name(g1, replace)
```

“lfit” permite agregar una línea de regresión entre las dos variables;
“scatter” indica la modalidad del gráfico;

El Gráfico 2 exhibe un patrón de asociación negativo entre dos variables, en este caso, entre la mortalidad infantil y la alfabetización de las mujeres, a partir de datos de 162 países provenientes de UNICEF. El coeficiente de correlación es -0,80.

Gráfico 2



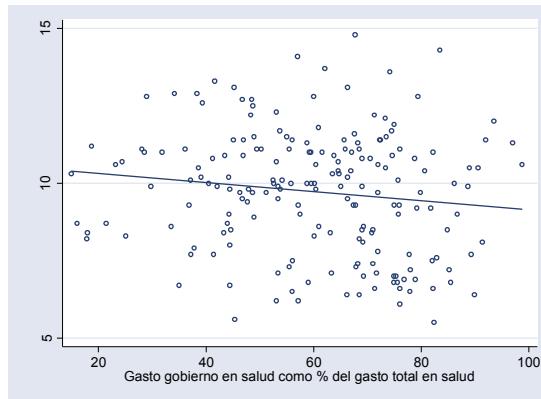
Nota:

```
use unicef, clear

graph twoway (lfit mort5_1999 literacy_f) (scatter mort5_1999 literacy_f, mstyle(p1)
ms(oh)), ytitle(Mortalidad Infantil) xtitle(Alfabetizacion) name(g2, replace)
legend(off)
```

Por otra parte, el Gráfico 3 es un ejemplo de la ausencia de relación evidente (o bien de relación débil) entre dos variables: en este caso se examina la asociación entre el gasto del gobierno en salud como proporción del gasto total en salud y la expectativa de años saludables perdidos. El coeficiente de correlación es de -0,13.

Gráfico 3

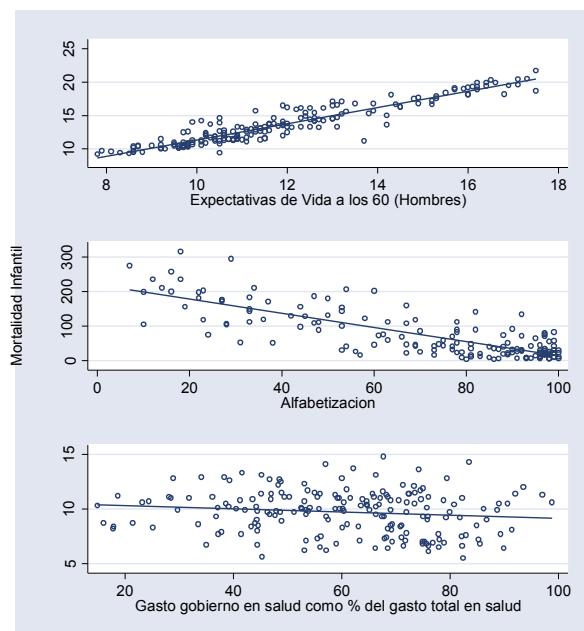


Nota:

```
use life, clear

graph twoway (lfit lhyb_m govexp) (scatter lhyb_m govexp, mstyle(p1) ms(oh)),
ytitle("Años saludables perdidos (expectativa)") xtitle("Gasto gobierno en salud
como % del gasto total en salud") name(g3, replace) legend(off)
```

Los tres gráficos anteriores se pueden poner juntos utilizando la siguiente instrucción: `graph combine g1 g2 g3, rows(3) ysize(6)`



Que el patrón de asociación encontrado, más o menos fuerte, no indica una relación de causalidad es algo claro. Por ejemplo, en el gráfico 1, la expectativa de vida de las mujeres no está causada, probablemente, por la expectativa de vida de los hombres. Y aún cuando pueda haber razones teóricas para pensar que hay una influencia causal de la alfabetización sobre la mortalidad infantil, ni el gráfico ni la recta de regresión pueden probar esta relación.

Entonces, si bien el **análisis de correlación** nos sirve para ver el grado de asociación lineal entre dos variables, este no indica una causalidad entre ellas. Por esta razón, cuando estamos interesados en causalidad de una o más variables sobre una variable de interés, utilizamos el **análisis de regresión**.

Factores de expansión (ponderadores)

Cuando trabajamos con muestras de datos que representan una población, la muestra generalmente es obtenida de un diseño muestral apropiado para lograr la representatividad de la población. Por lo tanto, cuando trabajemos con bases de datos que corresponden a una muestra de una población lo correcto es que además de las variables de interés se nos entregase una variable que contiene el factor de expansión, este factor de expansión nos indica cuantas observaciones de la población representa cada observación de la muestra. Luego, para hacer análisis a nivel poblacional debemos utilizar ese factor de expansión.

El comando `expand` de STATA lo que hace es expandir la base de datos de acuerdo a cierta variable, el factor de expansión en este caso. Supongamos un ejemplo sencillo donde tenemos tres observaciones:

The screenshot shows the STATA Data Editor window. At the top, there is a toolbar with buttons for 'Preserve', 'Restore', 'Sort', 'Hide', and 'Delete...'. Below the toolbar, a status bar displays the text 'weight[1] = 2'. The main area is a data grid with the following structure:

	folio	esc	ingreso	weight	
1	1	8	200000	2	
2	2	10	135600	3	
3	3	3	80000	1	

La variable weight es el factor de expansión e indica cuantas observaciones de la población representa cada observación de la muestra. Luego de ampliar la muestra a la población se puede utilizar el comando expand de la siguiente forma:

```
expand weight  
(3 observations created)
```

Obteniendo el siguiente resultado:

	folio	esc	ingreso	weight	
1	1	8	200000	2	
2	2	10	135600	3	
3	3	3	80000	1	
4	1	8	200000	2	
5	2	10	135600	3	
6	2	10	135600	3	

Sin embargo, en la mayoría de los casos esto no es óptimo ya que si por ejemplo trabajamos con una muestra de representatividad nacional como la encuesta CASEN, expandir la base de datos nos dejaría con cerca de 16 millones de observaciones, lo que hace inmanejable la base de datos en STATA. Por esta razón se utilizan los comandos de factores de expansión de STATA, que no expanden la base de datos pero para calcular las estadísticas descriptivas, hacer tabulaciones, regresiones, etc., es como si la ampliara.

En STATA existen cuatro tipos de factores de expansión que pueden ser utilizados en los distintos comandos, pero no todos los comandos son compatibles con todos los factores de expansión.

Frequency weights (fweights): son números enteros que indican el número de veces en que la observación es efectivamente observada. Este factor se utiliza cuando la base de datos ha sido condensada (collapsed) y contiene una variable que indica la frecuencia con que cada registro ocurre. Es un error

frecuente utilizar el frequency weight como sampling weight, este error sólo tiene consecuencias en la varianza de las estimaciones, valores p, y errores estándar.

En los comandos que se puede utilizar este factor de expansión se debe poner luego del comando y antes de la coma en caso que se utilice de la siguiente forma:

[fw=weight]

Samplig weights (pweights): en la mayoría de las encuestas donde la muestral fue extraída mediante un proceso de muestreo aleatorio dispondremos de un ponderador de muestreo (o sampling weight). Al usar esta opción de ponderador [pw=name] STATA utiliza el valor en el factor de expansión como el número de sujetos en la población que representa cada observación al computar proporciones, promedios, y parámetros de regresión. Además calcula varianzas robustas de manera tal que las estimaciones, de varianzas, errores estándar e intervalos de confianza son correctas.

En los comandos que se puede utilizar este factor de expansión se debe poner luego del comando y antes de la coma en caso que se utilice de la siguiente forma:

[pw=weight]

Analytical weights (aweights): este tipo de factor de expansión es apropiado cuanto se trabaja con datos que representan promedios, y el ponderador en este caso contiene el número de observaciones con la cuál se calculó dicho promedio. No es correcto usar este ponderador como sampling weight ya que las formulas que utiliza aweights asume que mientras mayor es el valor del ponderador las medidas de las observaciones son más precisas, pero en términos de sampling weight una observación no tiene por qué ser más precisa que otra observación.

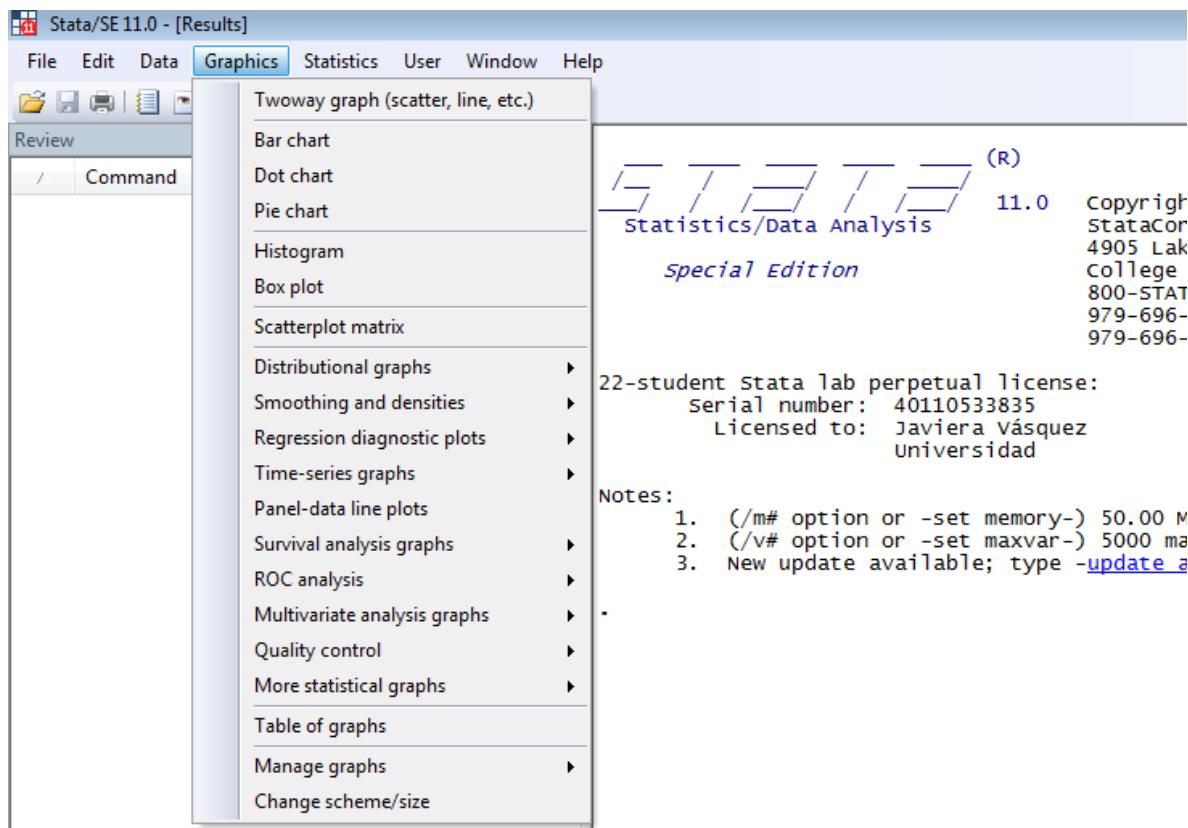
En los comandos que se puede utilizar este factor de expansión se debe poner luego del comando y antes de la coma en caso que se utilice de la siguiente forma:

[aw=weight]

Importance weights (iweights): este ponderador no tiene definición estadística formal. Puede ser usado por los programadores que necesitan implementar su propia técnica d análisis.

Capítulo XI: Gráficos

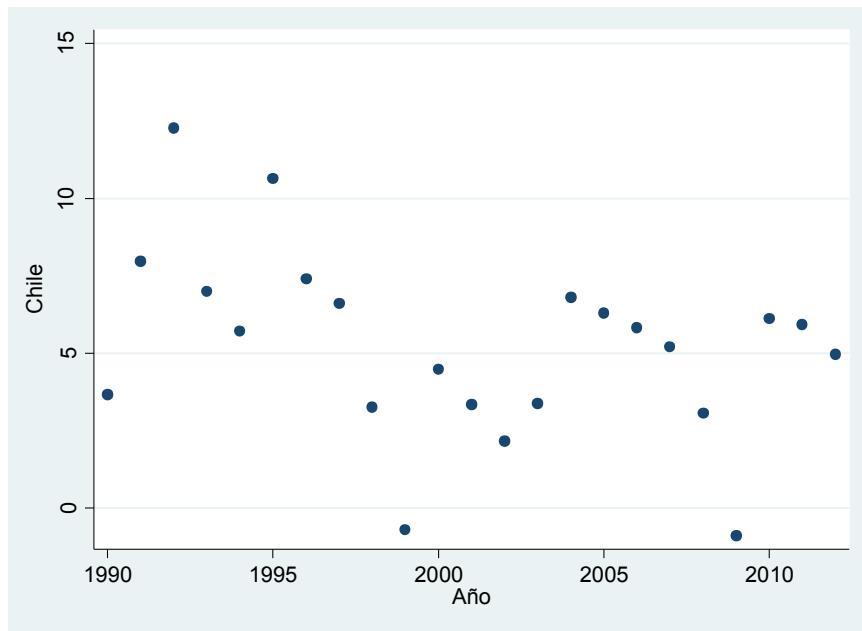
En STATA se pueden realizar gráficos de todos los tipos. Esto, al igual que otras funciones preestablecidas, no requiere del conocimiento de todos los comandos, ya que pinchando “Graphics” aparecen todas las opciones y en cada una de ellas un cuadro de opciones bastante completo:



1- Two way graph

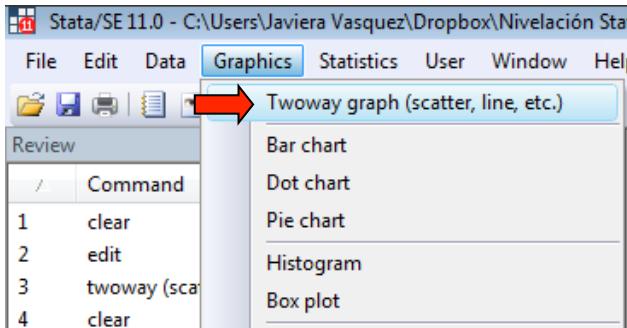
Estos tipos de gráficos se utilizan para analizar la relación que existe entre dos variables, por ejemplo el siguiente gráfico nos muestra cómo ha cambiado el crecimiento del PIB de Chile entre los años 1990 y 2012.

```
twoay (scatter chile año)
```

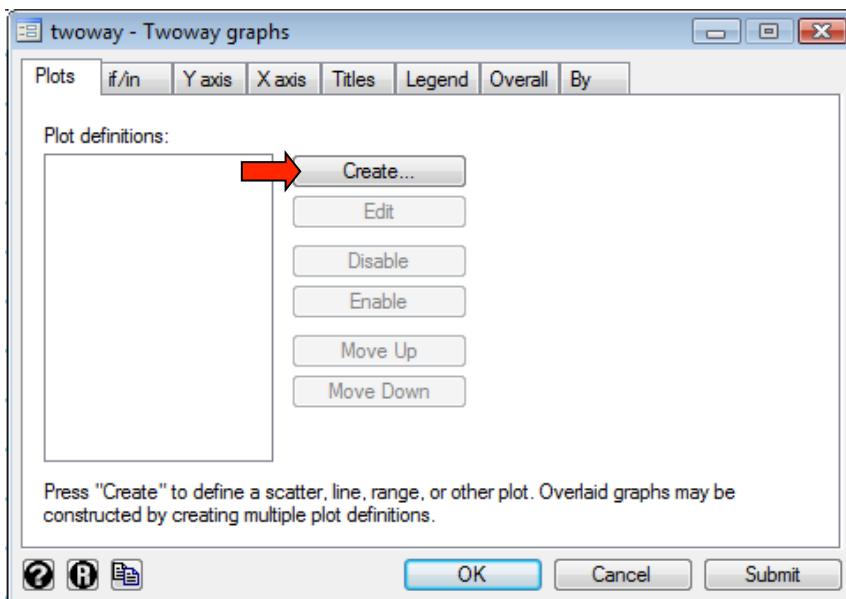


El comando para realizar un gráfico de dos variables que muestre la relación mediante puntos se llama `scatter`, primero se introduce la variable del eje vertical y luego la variable del eje horizontal.

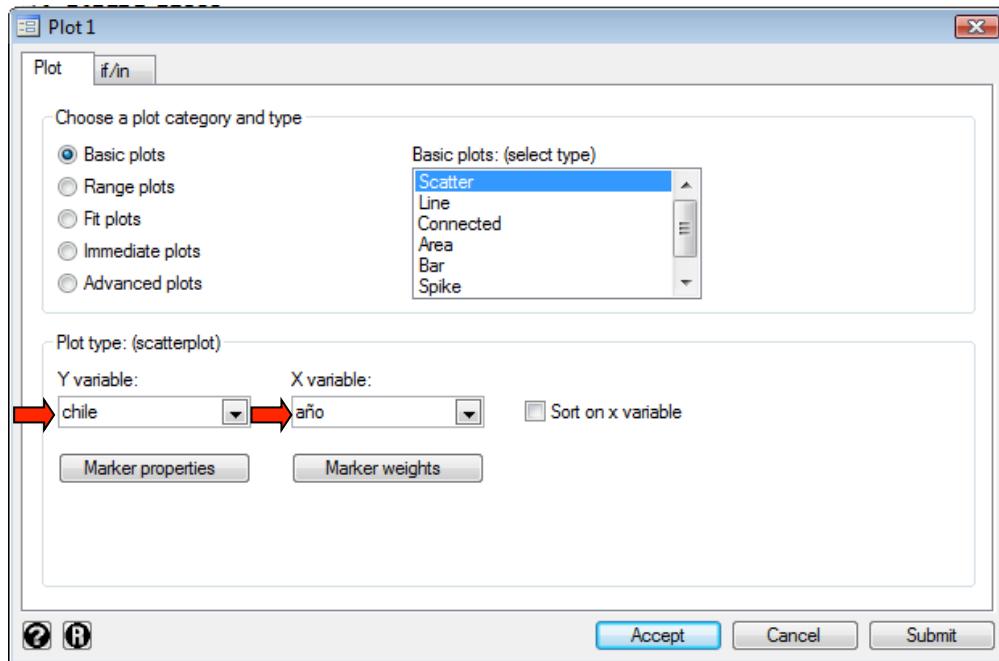
Este mismo gráfico lo podemos realizar siguiendo la opción de las ventanas de gráficos:



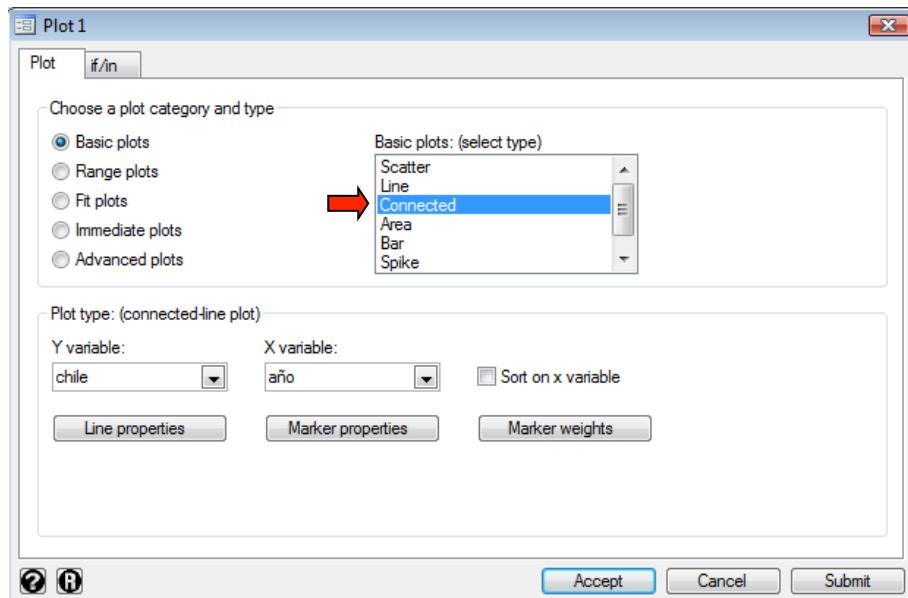
Un vez seleccionada la pestaña de “Twoway graph” aparecerá la siguiente ventana donde debemos pinchar “Create...”

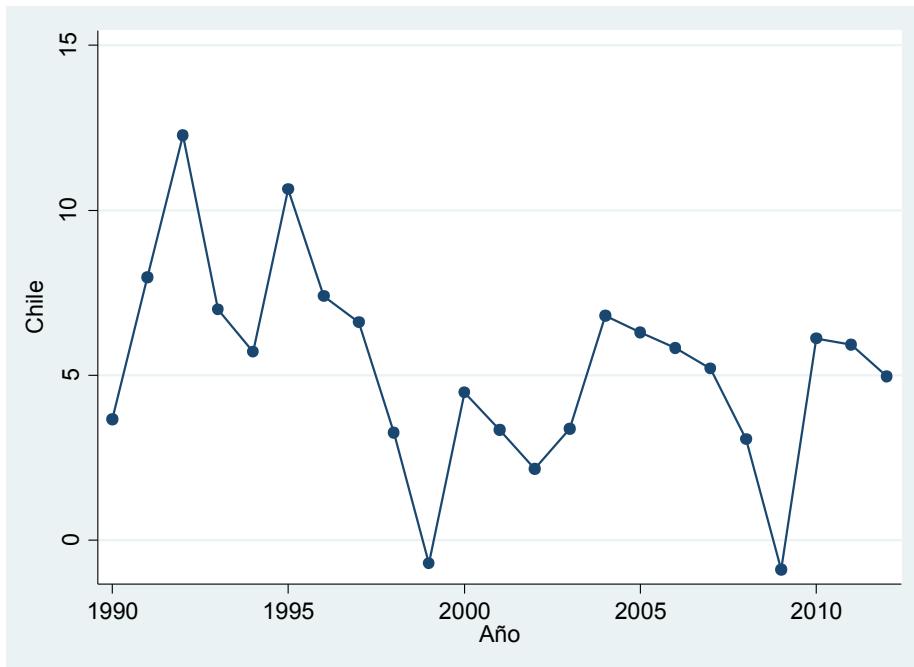


Aparecerá la siguiente ventana para crear el gráfico de acuerdo a las opciones que escogamos. Si seleccionamos “Basic Plots” el primero del listado es “scatter”, luego debemos introducir la variable correspondiente al eje vertical (Y) y la variable correspondiente al eje horizontal (X):



Luego, si aceptamos y luego damos OK, nos aparecerá el mismo gráfico anterior. También podemos elegir otros tipos de gráficos para mostrar la misma relación entre estas dos variables. Por ejemplo:

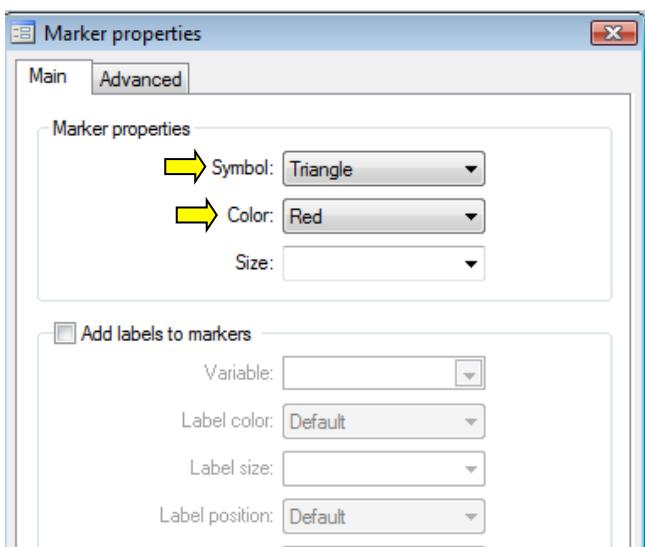
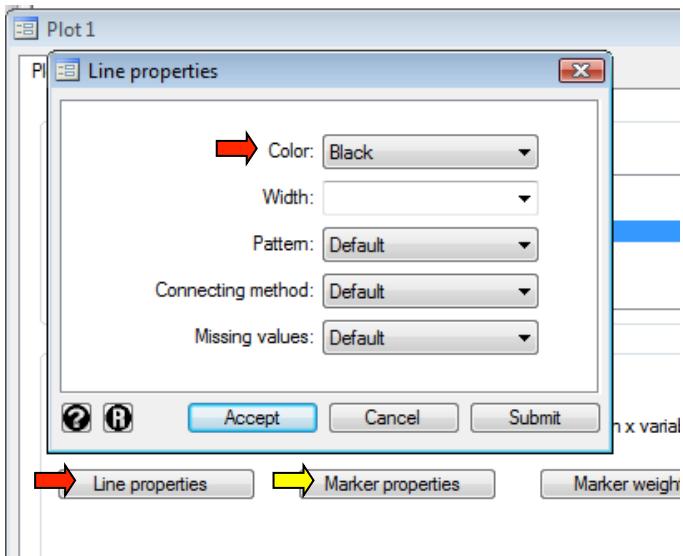




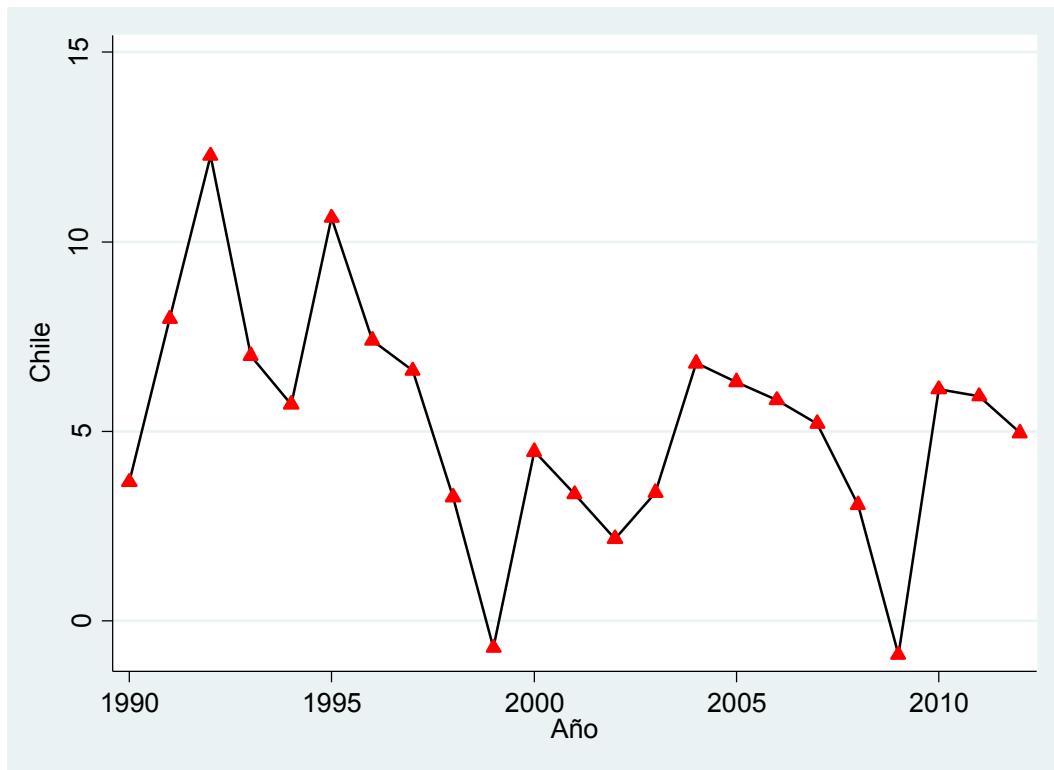
El comando para realizar el gráfico anterior es:

```
two way (connected chile año)
```

Luego podemos ocupar las distintas opciones de la ventana del gráfico para cambiar los colores, marcadores, cambiar los nombres y numeración de los ejes, etc.



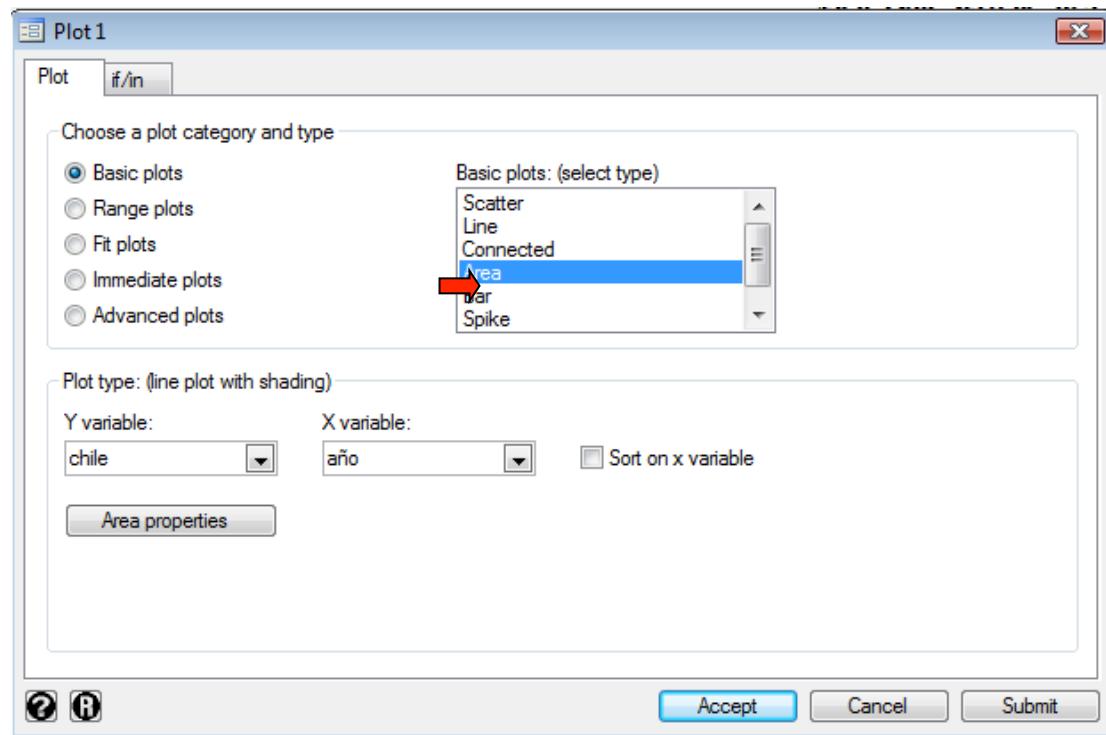
Con estas opciones el mismo gráfico quedaría de la siguiente manera:



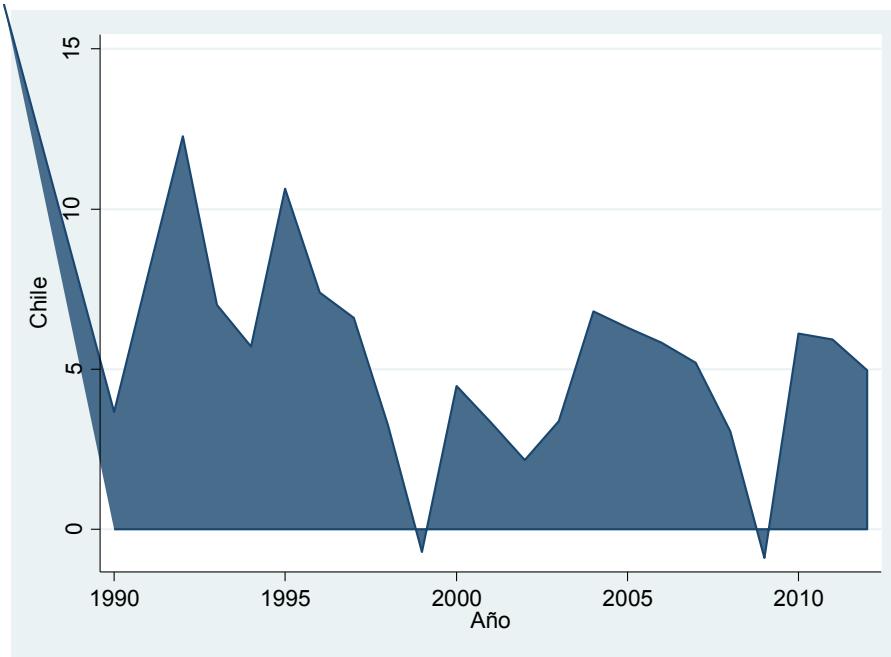
El comando para realizar el gráfico anterior es:

```
twoway (connected chile año, mcolor(red) msymbol(triangle)  
lcolor(black))
```

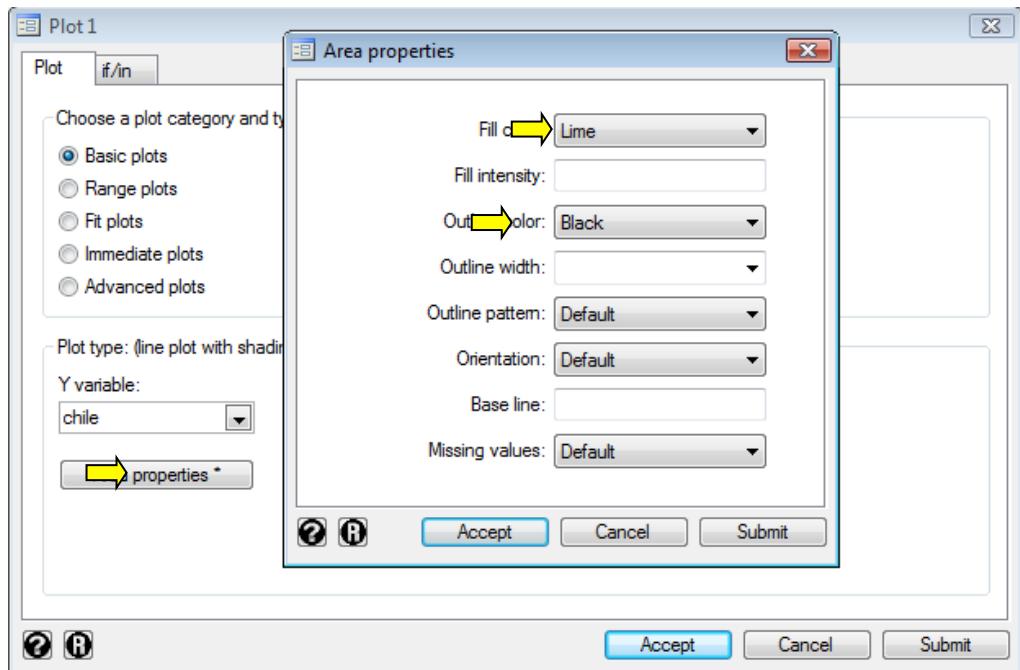
También podemos hacer un gráfico de área:



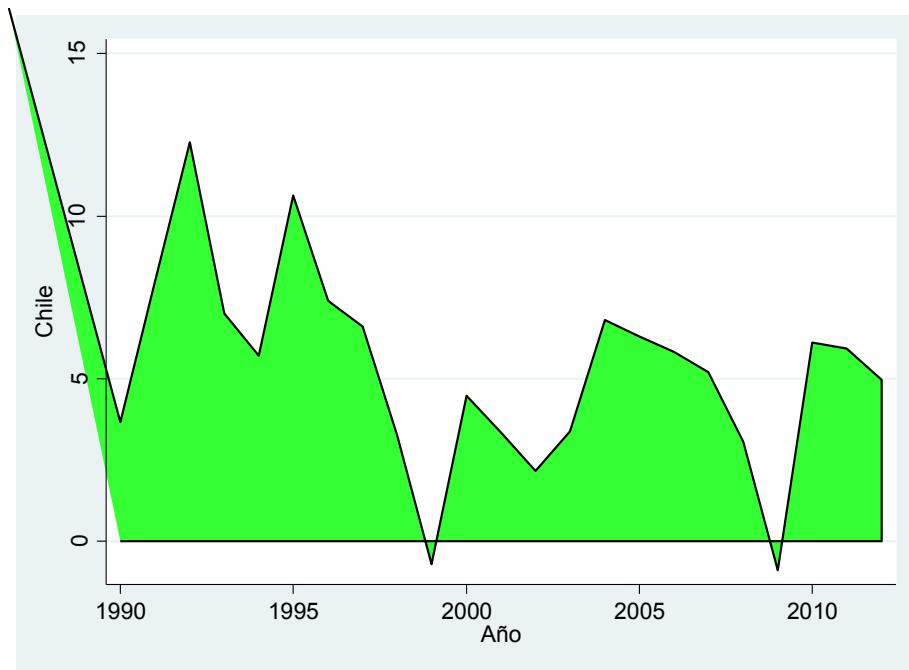
`twoway (area chile año)`



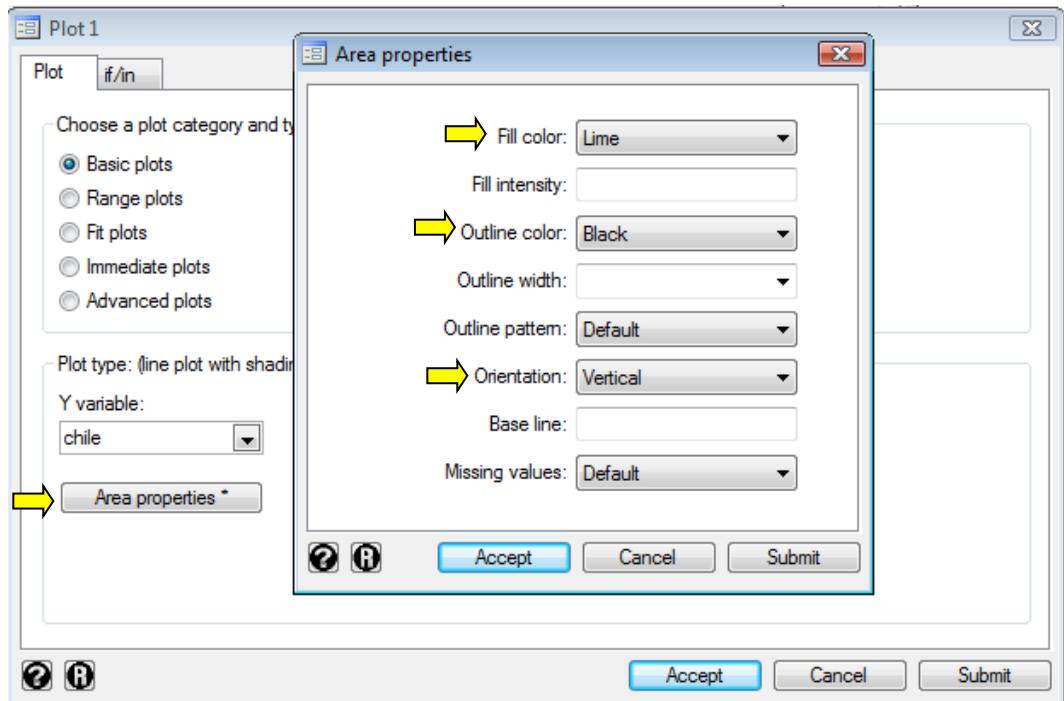
Donde también podemos elegir los colores del área y formato:



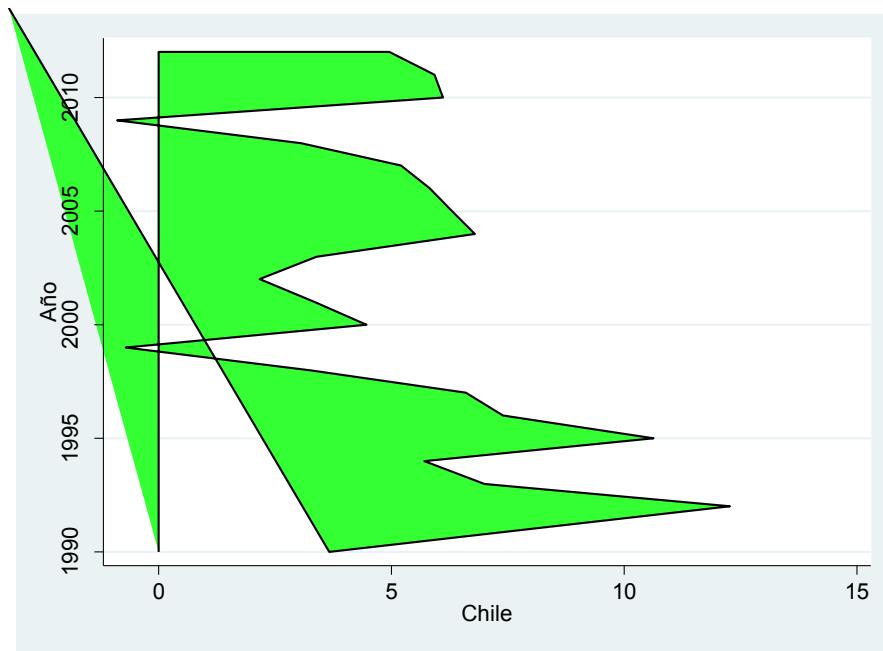
```
twoway (area chile año, fcolor(lime) lcolor(black))
```



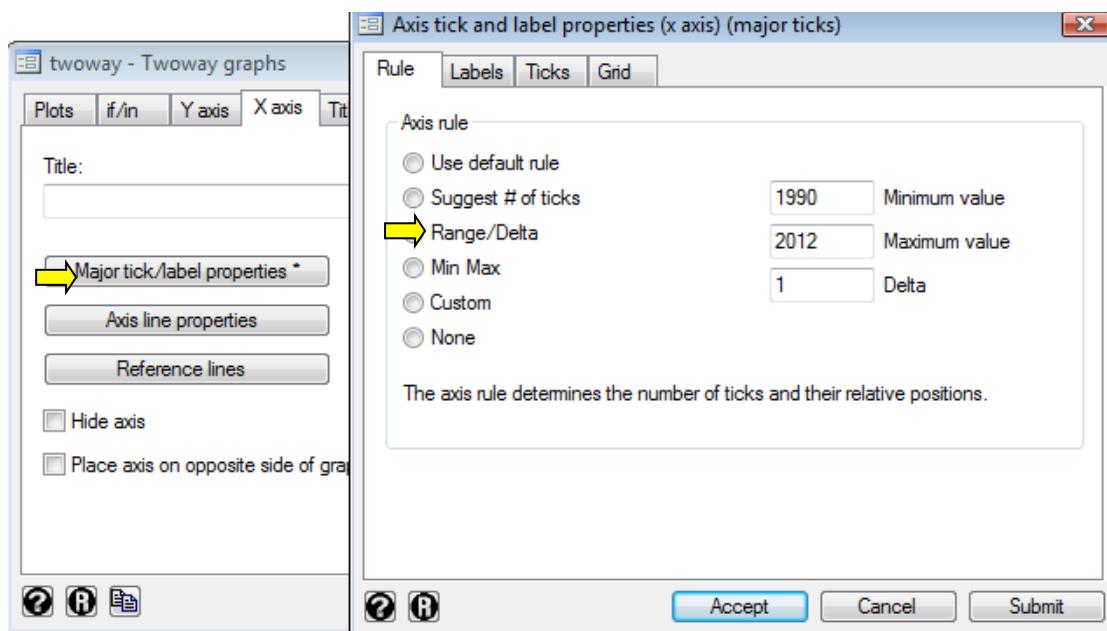
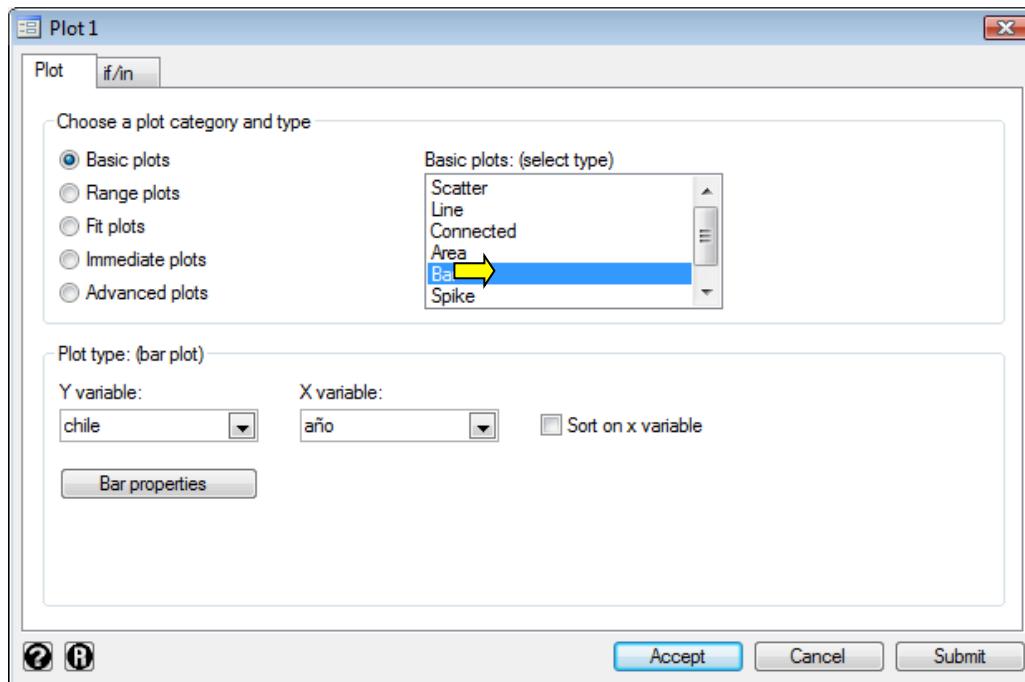
Y cambiar la orientación del gráfico:

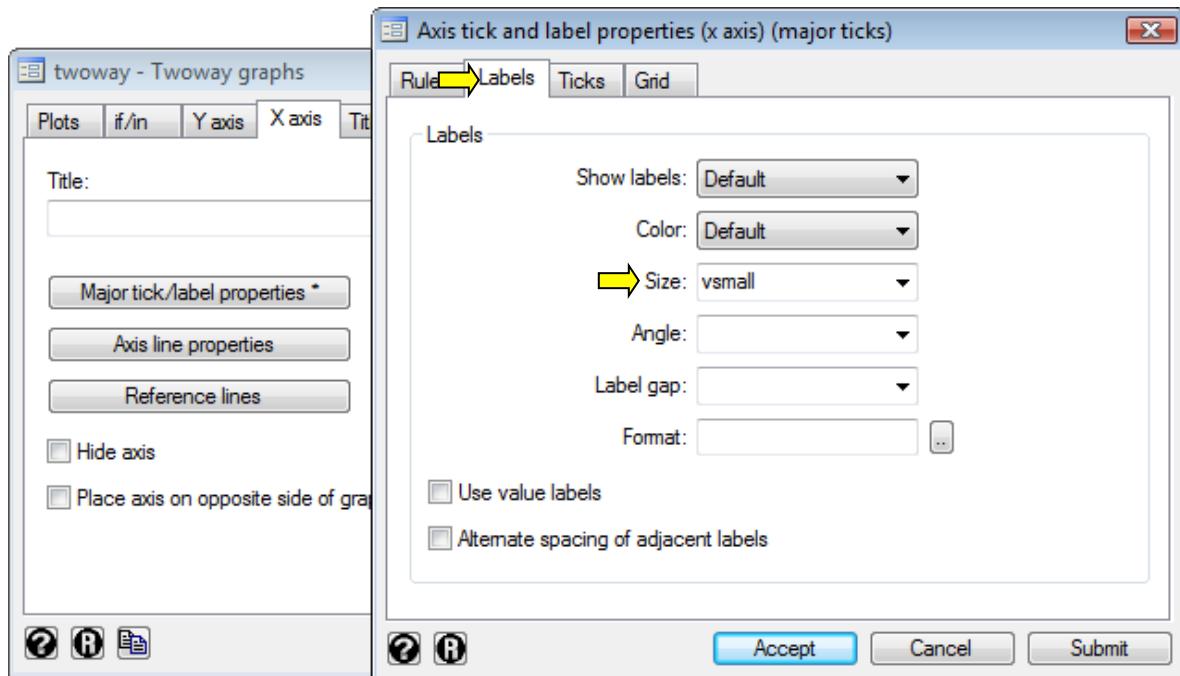


```
twoway (area chile año, fcolor(lime) lcolor(black) horizontal)
```

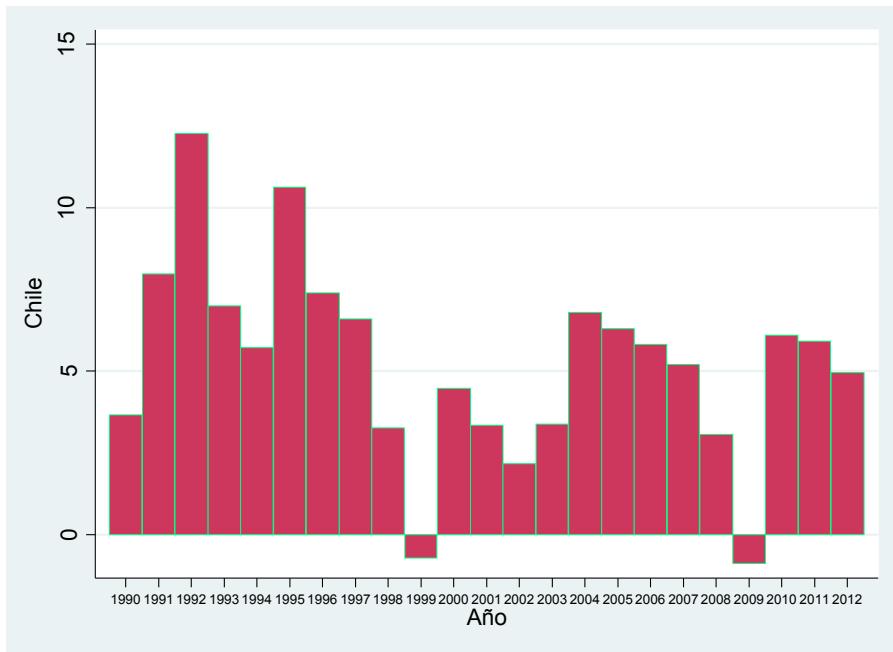


También se puede realizar el mismo gráfico anterior pero en formato de barras:

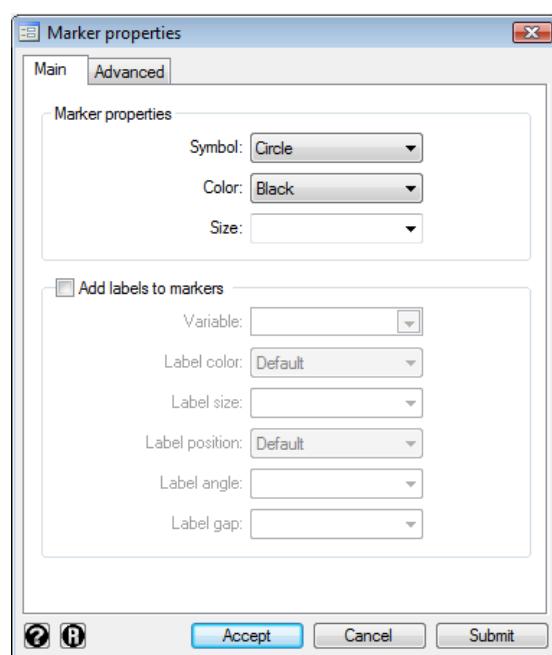
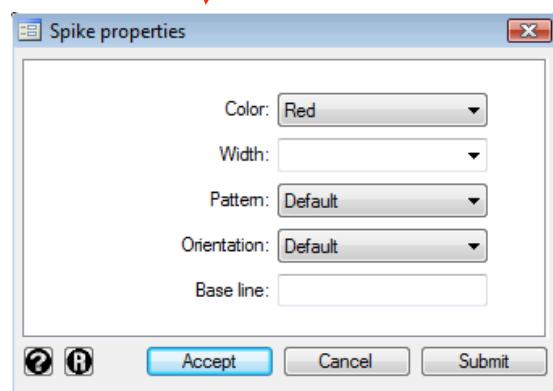
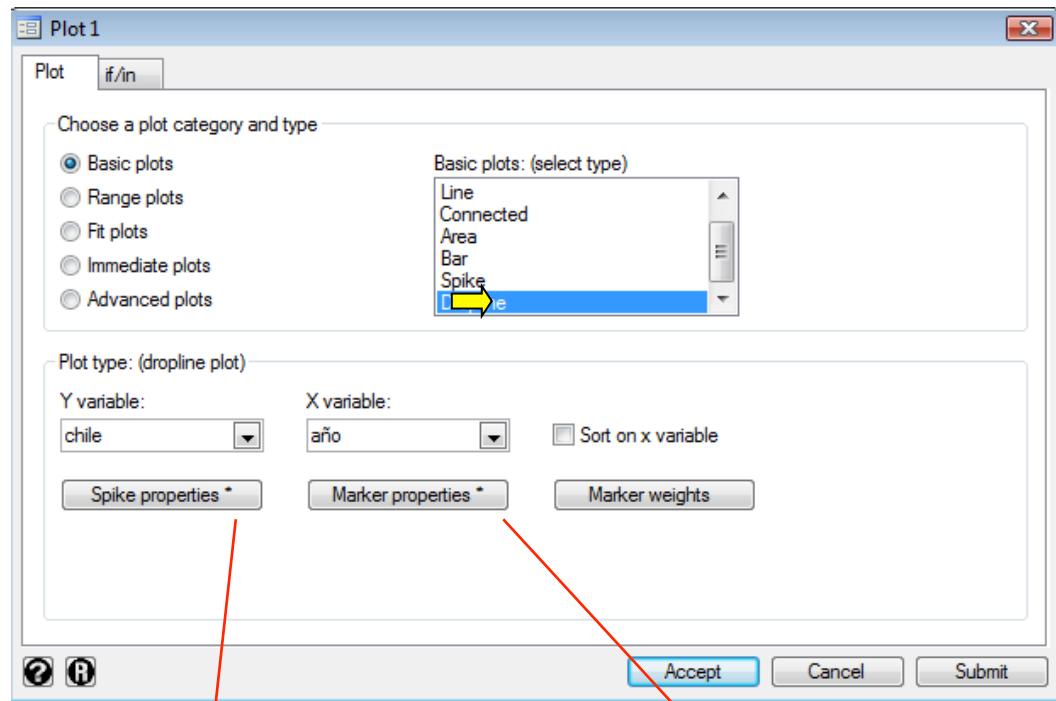




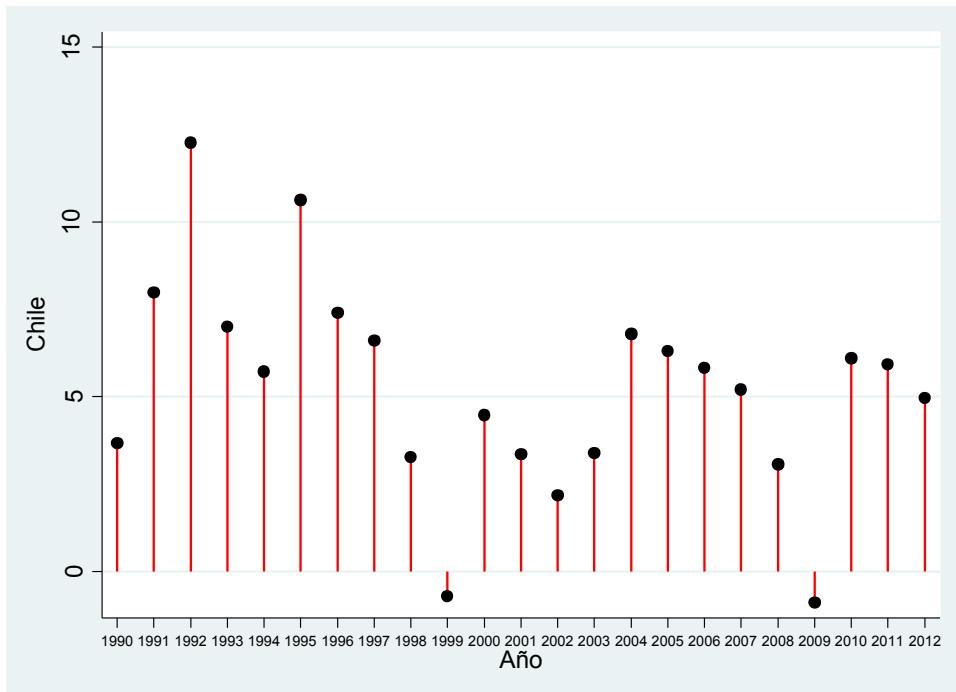
```
twoway (bar chile año, fcolor(cranberry) lcolor(mint)), xlabel(1990(1)2012, labsize(vsmall))
```



Finalmente, el mismo gráfico lo podemos hacer en formato "dropline"



```
twoway (dropline chile año, mcolor(black) msymbol(circle) lcolor(red)),
xlabel(1990(1)2012, labsize(vsmall))
```



Ejercicio:

Abra la base de datos pcobre.dta. Ud. Dispone de datos del precio del cobre desde Enero de 1987 a Marzo de 2013.

Genere una variable llamada "fecha" a través del siguiente comando de stata:

```
g fecha=ym(año,mes)
format fecha %tm
```

Realice un gráfico que muestre la evolución del precio del cobre a través de un gráfico de dos sentido que conecte los puntos de observación. Elija los colores de su preferencia, ponga título al gráfico y una nota que diga que la fuente es el Banco Central.

Ahora realice un gráfico de área pero solo utilizando las observaciones del año 2005 en adelante. Nuevamente edite el gráfico con sus colores, títulos, y notas.

Realice un gráfico de barras que muestre la evolución del precio del cobre

2- Bar Chart

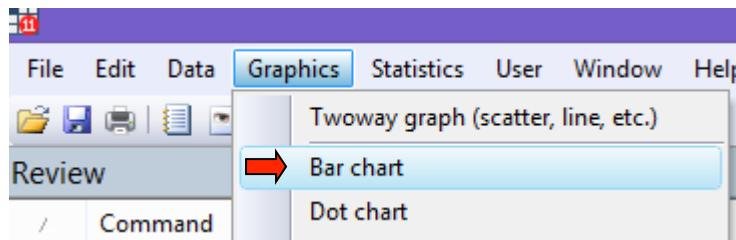
El gráfico de barras de STATA nos permite resumir de manera gráfica y comparativa estadísticas descriptivas, como la media, mediana, entre otras de una o más variables de interés.

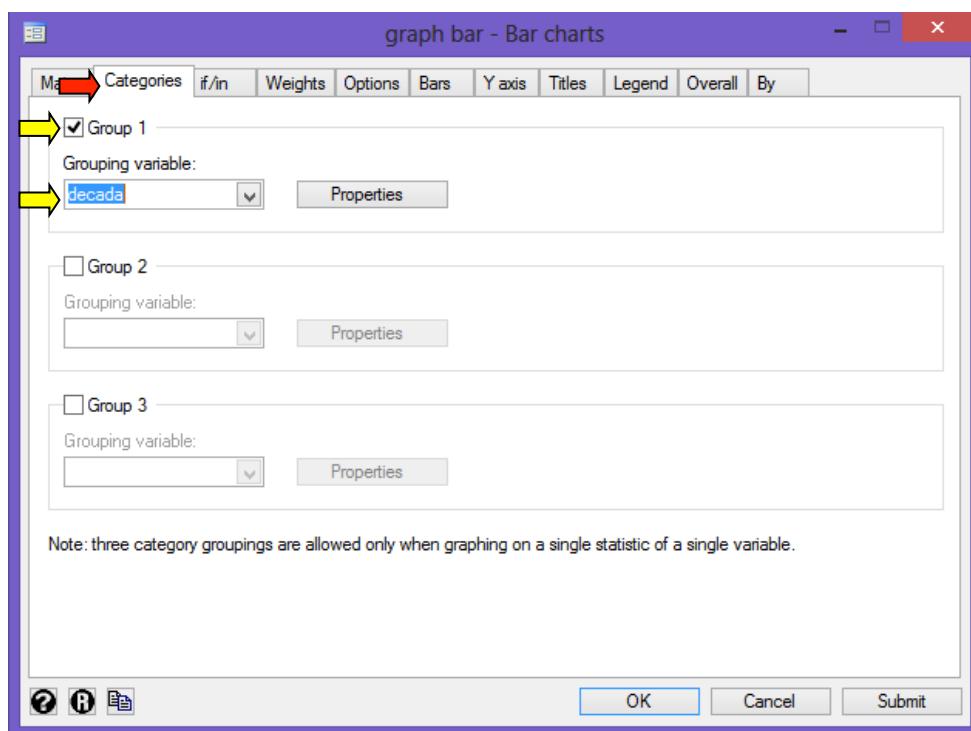
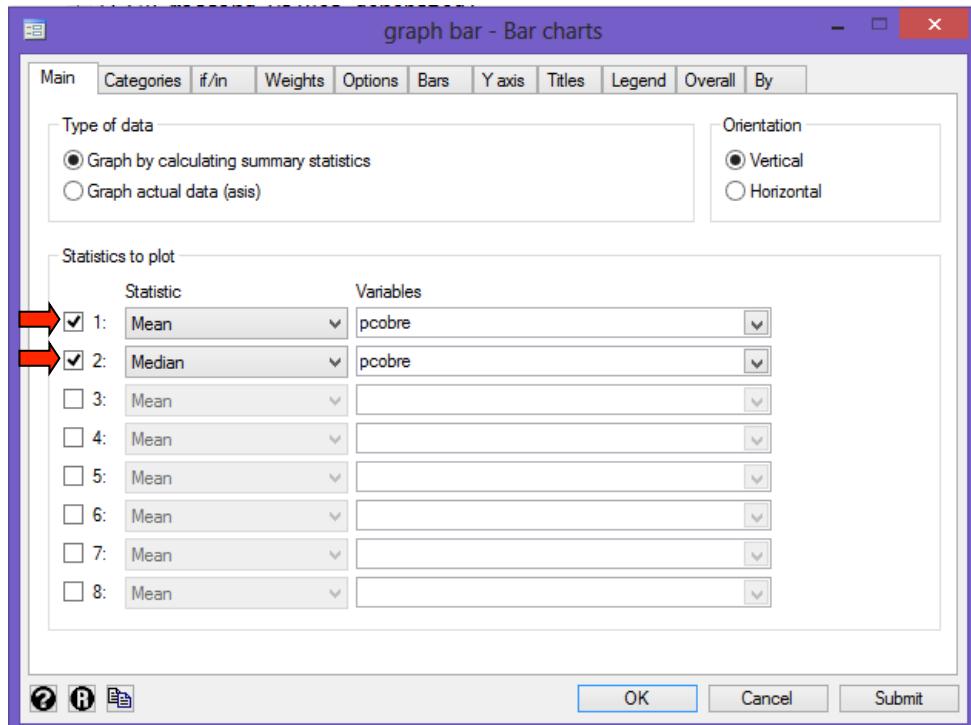
Por ejemplo, utilizando la misma base de datos del precio del cobre podemos hacer un gráfico que muestre el precio promedio y mediano para las distintas décadas.

Primero vamos a generar la variable década:

```
g decada=1 if año<1990  
replace decada=2 if año>=1990 & año<2000  
replace decada=3 if año>=2000 & año<2010  
replace decada=4 if año>=2010  
label define decada 1 "80" 2 "90" 3 "00" 4 "10"  
label values decada decada
```

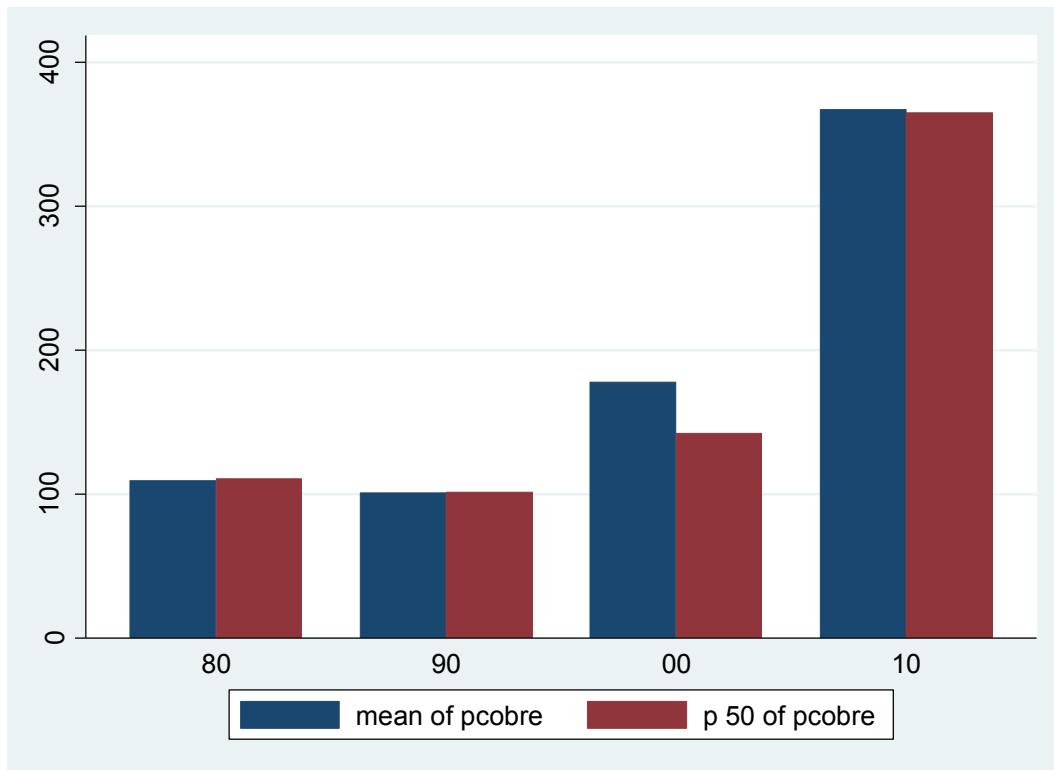
Ahora vamos a las opciones de gráfico y escogemos “gráfico de barras”:



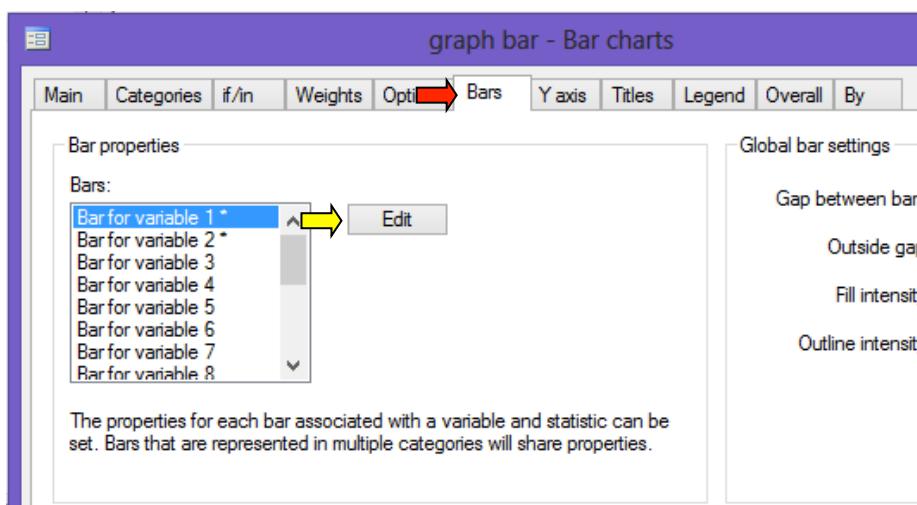


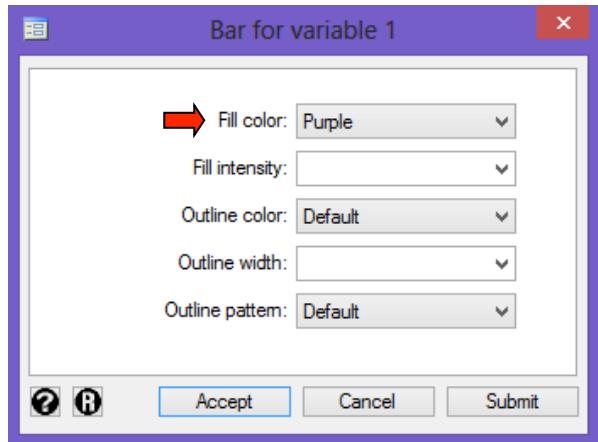
El gráfico resultante es el siguiente:

```
graph bar (mean) pcobre (median) pcobre, over(decada)
```



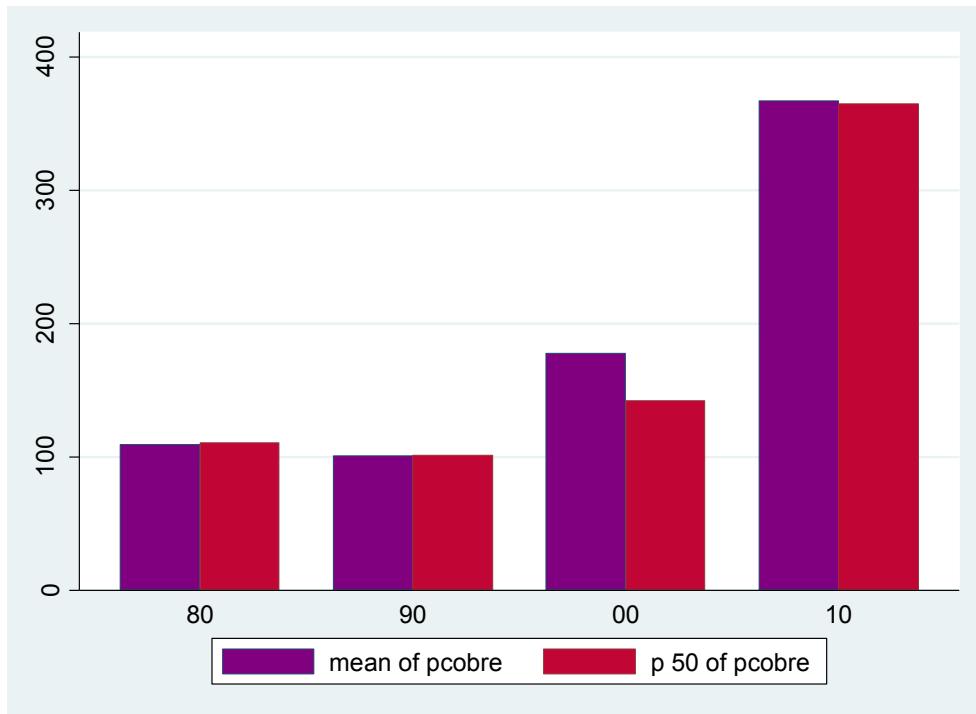
Al igual que antes podemos editar los colores del gráfico, poner un título e incluso cambiar la leyenda.



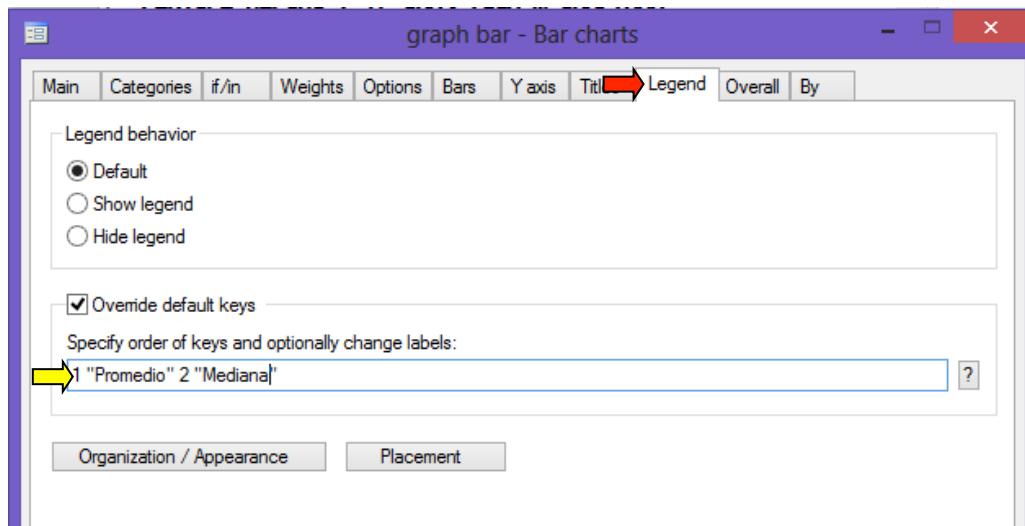


Obteniendo el siguiente gráfico:

```
graph bar (mean) pcobre (median) pcobre, over(decada) bar(1, fcolor(purple)) bar(2, fcolor(cranberry))
```

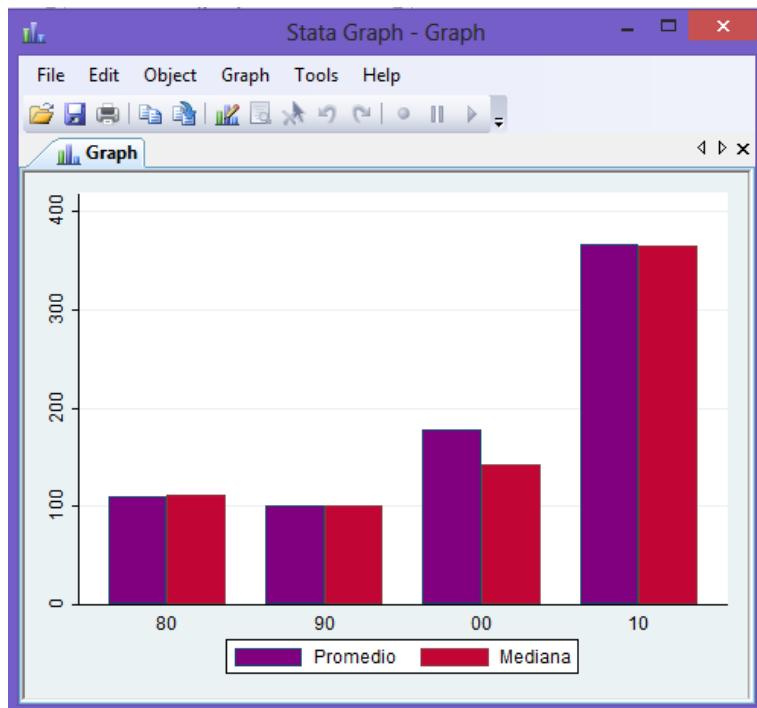


Podemos cambiar la leyenda de la siguiente manera:

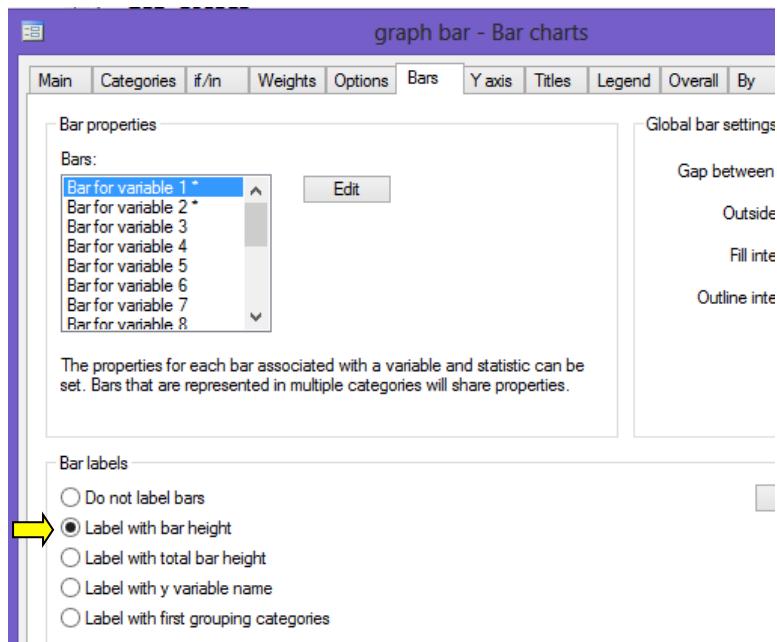


El gráfico resultante es el siguiente:

```
graph bar (mean) pcobre (median) pcobre, over(decada) bar(1, fcolor(purple)) bar(2, fcolor(cranberry)) legend(order(1 "Promedio" 2 "Mediana"))
```

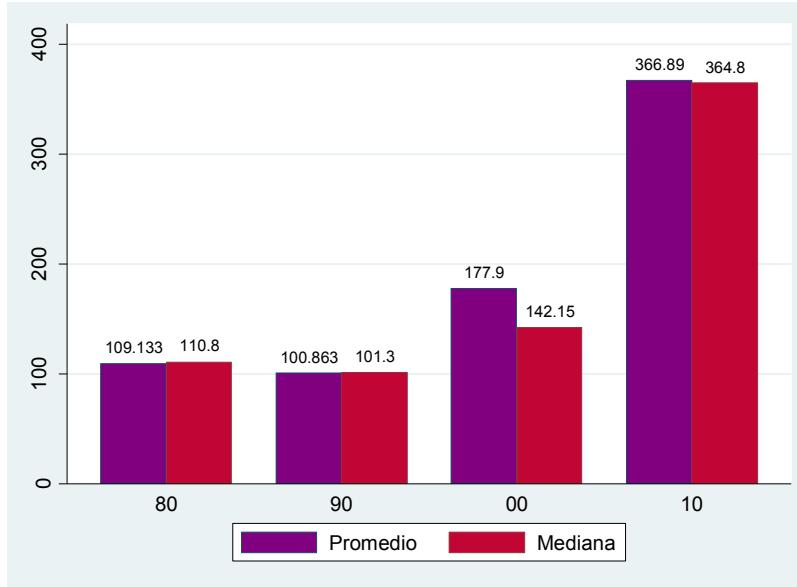


Finalmente, podemos también poner los valores correspondientes a cada una de las barras:



Así, el gráfico queda de la siguiente manera:

```
graph bar (mean) pcobre (median) pcobre, over(decada) bar(1, fcolor(purple)) bar(2, fcolor(cranberry)) blabel(bar) legend(order(1 "Promedio" 2 "Mediana"))
```



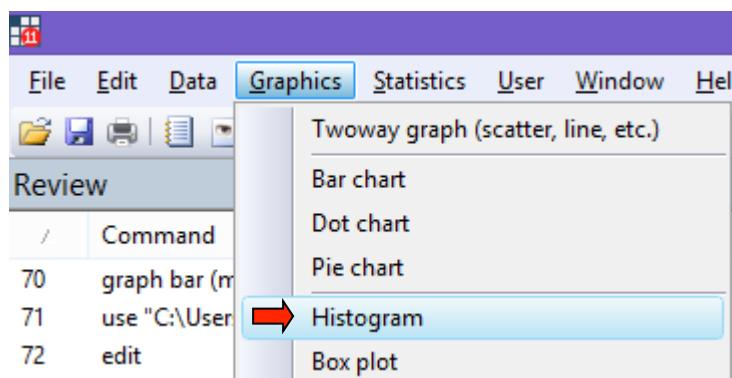
Ejercicio:

Utilizando la base de datos exteps09 realice un gráfico de barras donde cada una de las barras represente el ingreso mensual promedio de cada año de escolaridad (0 a 21).

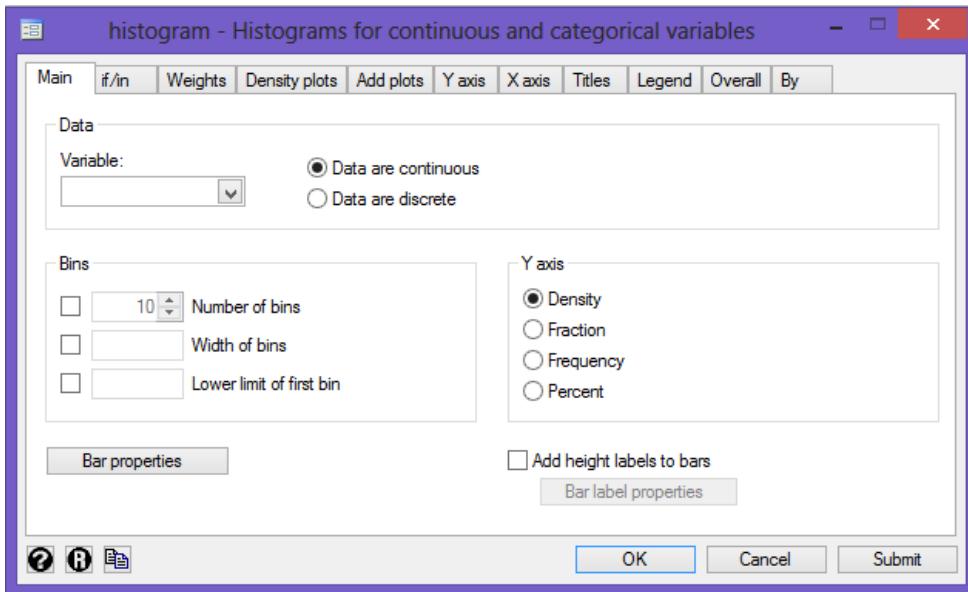
Luego haga el mismo gráfico pero separando las barras por hombre y mujer

3- Histogram

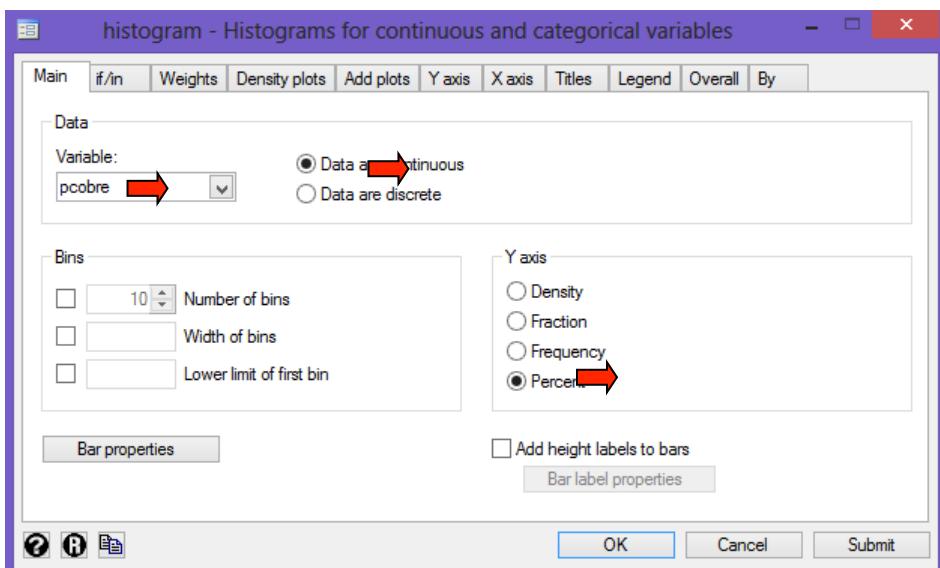
Un histograma es un tipo de gráfico que nos permite caracterizar como se distribuye la muestra o población de datos en los valores que puede tomar la variable. Para hacer un histograma debemos elegir la opción “Histogram” de los gráficos.



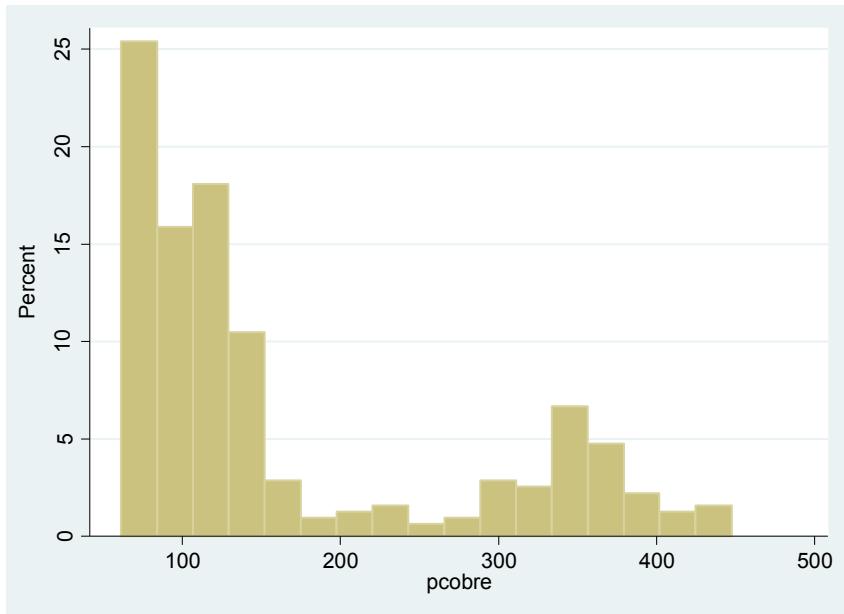
Donde nos aparecerá la siguiente ventana:



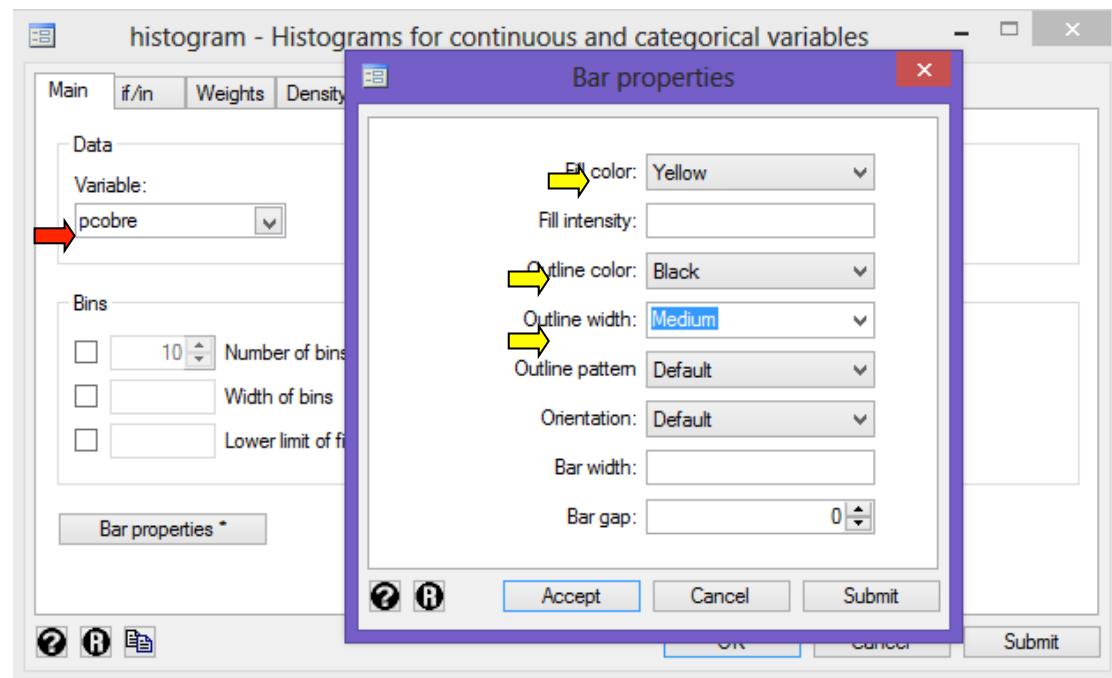
Aquí debemos introducir la variable sobre la cuál queremos construir el histograma, indicar si la variable es continua o discreta, que queremos que muestre en el eje vertical (densidad, fracción, frecuencia o porcentaje) entre otras opciones. Por ejemplo, a continuación se presenta el histograma del precio del cobre:



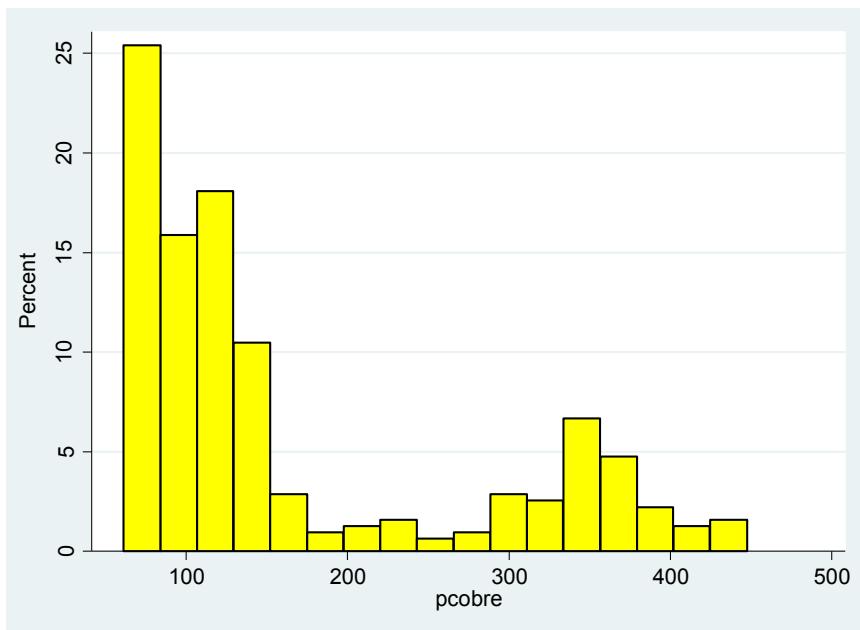
histogram pcobre, percent



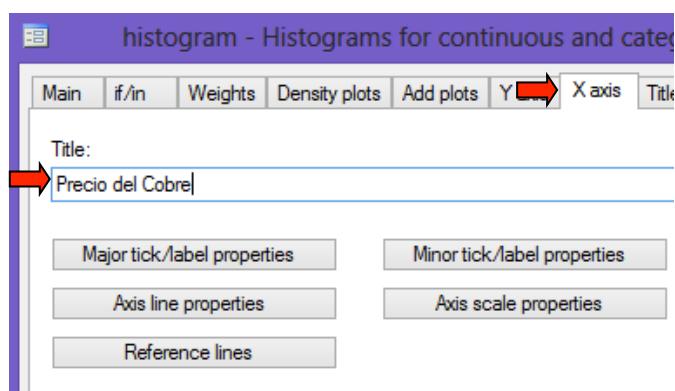
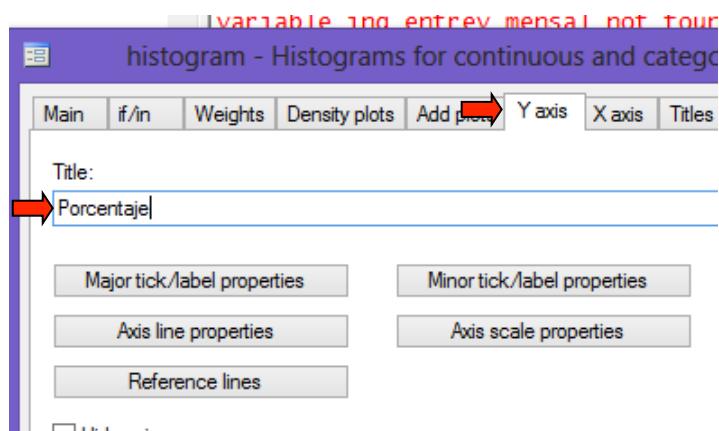
Al igual que en los gráficos anteriores podemos modificar el grafico con las opciones de colores y títulos.

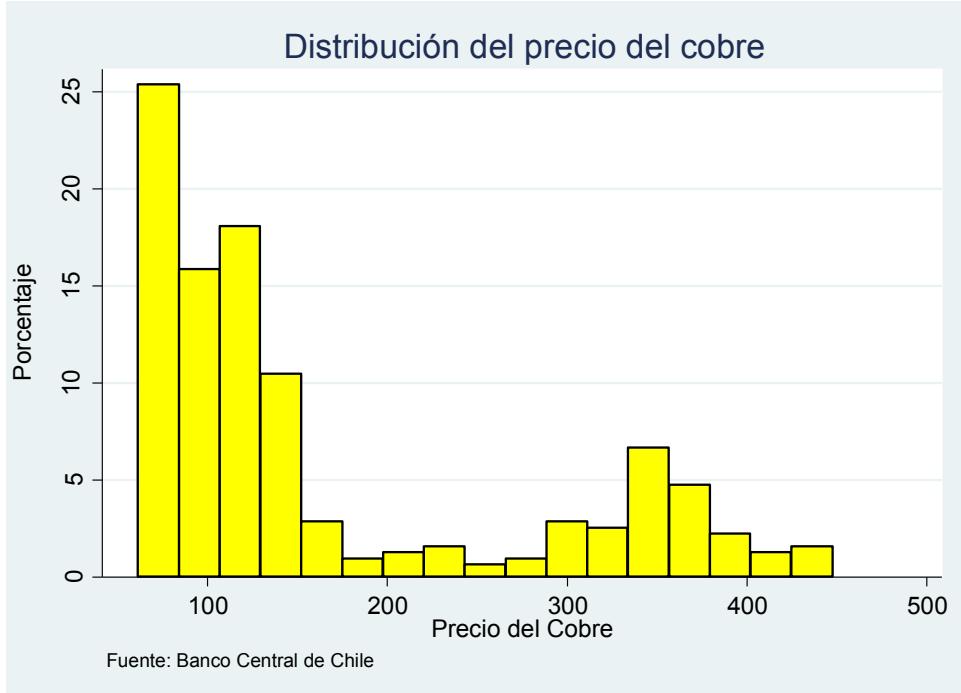
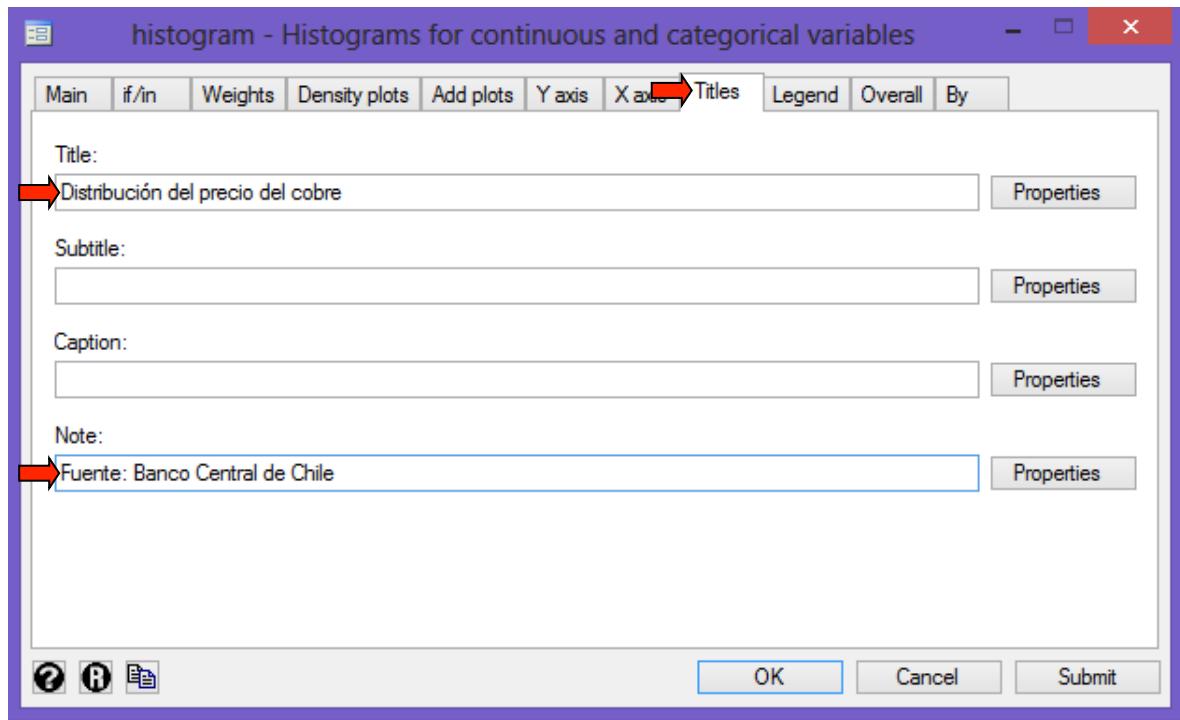


```
histogram      pcobre,      percent      fcolor(yellow)      lcolor(black)
lwidth(medium)
```



Para poner título al gráfico, cambiar el nombre de los ejes y poner un nota debemos hacer lo siguiente:



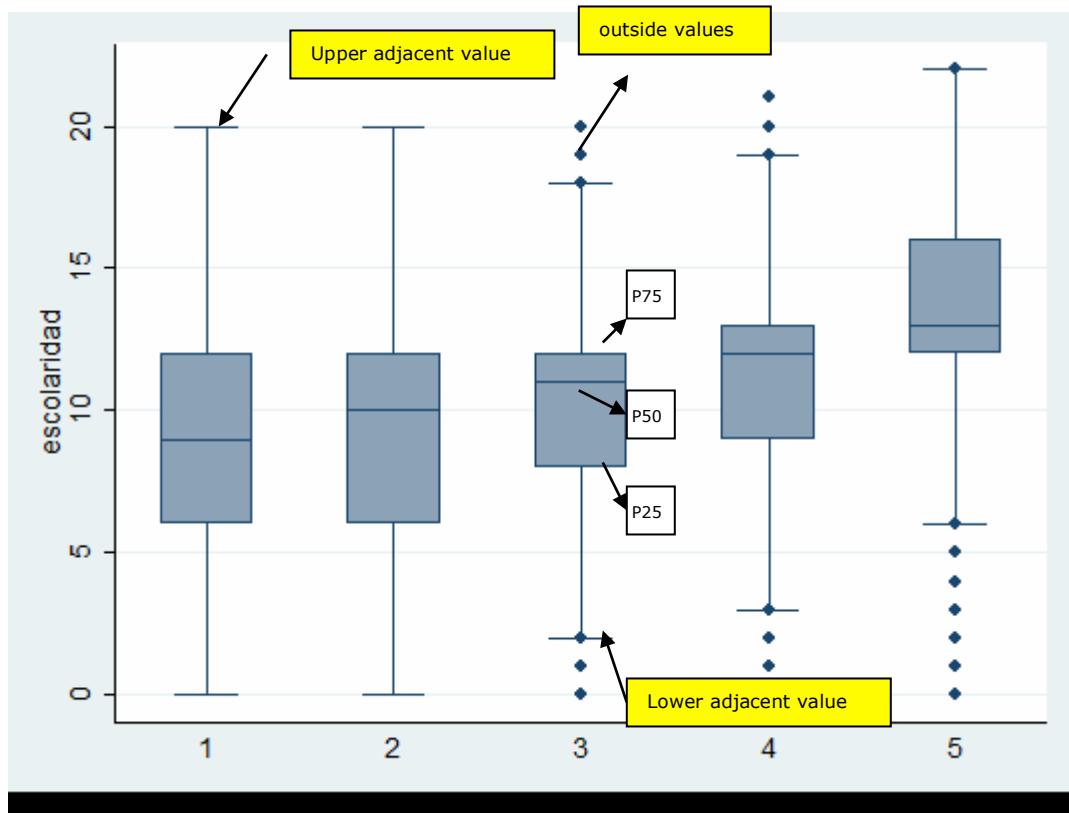


Ejercicio: utilizando la base de datos exteps09.dta realice un histograma de la variable edad. Edite el gráfico con títulos y sus preferencias, y adhiera una línea de distribución normal.

4- Box plot

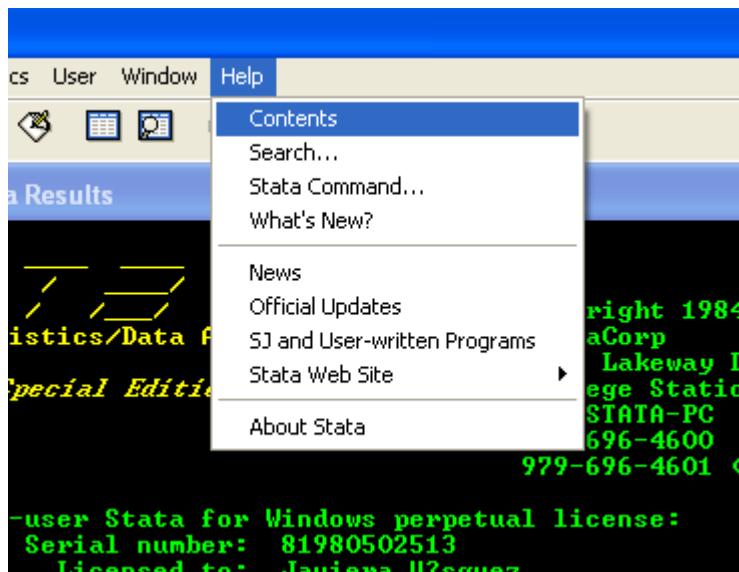
Este tipo de gráfico de STATA es bastante útil ya que nos permite resumir algunos de los indicadores de dispersión y tendencia central antes descritos. Este gráfico entre el siguiente dibujo como resultado:

```
graph box esc [w=expr], over(quintil)
```

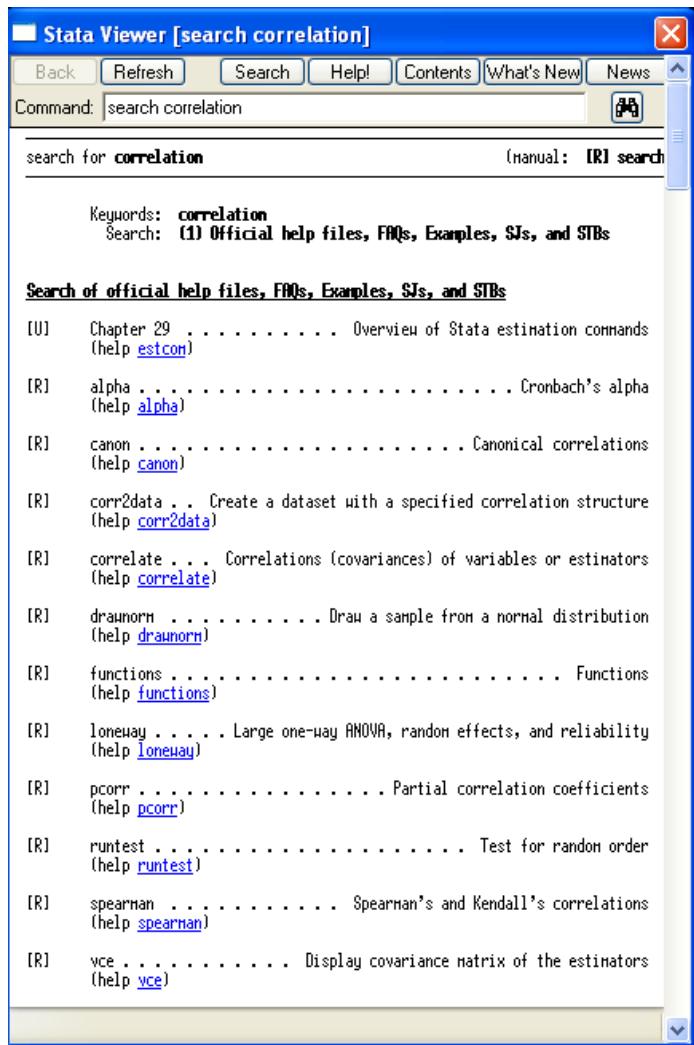


Capítulo XII: Ayuda: Help

La ayuda que trae este programa es bastante amigable y fácil de utilizar. Si Ud. Se dirige a "Help" se despliega un cuadro con diferentes opciones, puede buscar ayuda por contenidos, por comandos, etc..., en "What's new" encontrará todas las novedades del programa, que nuevos comandos hay y podrá descargarlos haciendo un update.



Por ejemplo, supongamos se quiere saber como calcular correlación entre variables. Entonces se debe buscar en "Search":



Dependiendo que este buscando pincha el comando (en azul), y se abrirá una ventana con una completa ayuda sobre el comando.

Viewer (#1) [help correlate]

Back Refresh Search Help Contents What's New News

Command: help correlate

help correlate, help pwcorr dialogs: **correlate pwcorr**

Title

[R] **correlate** — Correlations (covariances) of variables or estimators

Syntax

Display correlation matrix or covariance matrix

correlate [*varlist*] [*if*] [*in*] [*weight*] [, *correlate_options*]

Display all pairwise correlation coefficients

pwcorr [*varlist*] [*if*] [*in*] [*weight*] [, *pwcorr_options*]

correlate_options description

Options

_coef	display correlation or covariance matrix of the coefficients
means	display means, standard deviations, minimums, and maximums with matrix
nofomat	ignore display format associated with variables
covariance	display covariances
wrap	allow wide matrices to wrap

pwcorr_options description

Main

obs	print number of observations for each entry
sig	print significance level for each entry
print(#)	significance level for displaying coefficients
star(#)	significance level for displaying with a star
bonferroni	use Bonferroni-adjusted significance level
sidak	use Sidak-adjusted significance level

varlist may contain time-series operators; see **tsvarlist**.
by may be used with the **correlate** and **pwcorr**; see **by**.
aweights and *fweights* are allowed; see **weight**.

Description

The **correlate** command displays the correlation matrix or covariance matrix for a group of variables or for the coefficients of the most recent estimation. If *varlist* and **_coef** are not specified, the matrix is displayed for all variables in the data. Observations are excluded from the calculation due to missing values on a casewise basis. Also see **estat**.

Para que el Help de STATA le sea realmente útil es fundamental que Ud. aprenda a leer la sintaxis de cada comando. Por ejemplo, en este caso la sintaxis del comando correlate es la siguiente:

`correlate [varlist] [if] [in] [weight] [, correlate_options]`

Esta sintaxis nos indica que el comando correlate puede ser utilizado entregando el listado de variables, sino se entrega el default es considerar todas las variables en la base de datos. Se puede utilizar la opción if, in y weight todas ellas antes de la coma, la que es utilizada también en forma opcional para poner algunas opciones del comando. Entonces siempre lo que aparece en paréntesis cuadrados son alternativas opcionales del comando. Más adelante de la sintaxis, se indican cuales son estas "correlate_options", también se presenta una descripción general del comando y sus opciones.

A continuación se muestra otro ejemplo de sintaxis, para el comando `table`:

table rowvar [colvar [supercolvar]] [if] [in] [weight] [, options]

Recordemos que todo lo que esta entre paréntesis cuadrado son opciones voluntarias de comando. En este caso rowvar no esta entre paréntesis cuadrado, significa que debemos obligadamente señalar la variable fila en la tabulación, es voluntario indicar la variable de columna, si no se indica la tabulación resultante es una distribución de la variable sola, si se indica colvar la tabla será un cruce entre rowvar y colvar. Este ejemplo nos permite notar la diferencia en aquellos insumos del comando que son obligatorios de los que son voluntarios.

Puede ser que algunos de los comandos que aparezcan Ud. no los tenga cargados, en este caso si se encuentra conectado a Internet los podrá bajar online. Aparecerá algo de la siguiente forma:

SJ-3-3 [snp15_4](#) Software updates for **sonersd**
(help **sonersd** if installed) R. Neuson
Q3/03 SJ 3(3):325
bug fix for **sonersd**

Donde pinchando la palabra en azul se puede descargar el comando, así como su ayuda.

Una forma más completa de buscar ayuda es a través de los manuales, estos además de traer una ayuda sobre los comandos generalmente contiene toda una descripción teórica de ellos. Por ejemplo, en el caso de un modelo de regresión lineal, explica lo que es una modelo de este tipo.

Además el manual de gráficos es muy útil, ya que trae múltiples ejemplos que se pueden adecuar a los que uno anda buscando.

En esta parte veremos un ejemplo como el visto en la parte 3 y luego con la misma base de datos realizaremos varios tipos o formas de gráficos.