**Notas de clase – Taller de Stata**[1]

**Clase 6 – Escalares, Vectores y Matrices**

## Contenido





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

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

## 1. Escalares

*Stata* es un programa que permite almacenar números como escalares y operar matrices. En un escalar se puede almacenar un número o una cadena de caracteres, que puede contener varias palabras, por medio del comando scalar. A continuación, se presentan algunos ejemplos sobre el tipo de información que puede ser guardado en escalares:

Ej. 1 ⟶ scalar *a* = "La raíz cuadrada de dos es "
⟶ scalar *b* =sqrt(2)
⟶ display *a b*

Los escalares son bastante útiles para almacenar los resultados de una estimación la ejecución de los diferentes comandos. En la ayuda de cada comando es posible saber qué información se puede guardar como escalares de ejecutarlo. Por ejemplo, el comando sum guarda como escalares el número de observaciones, la media, la desviación estándar el valor máximo, el valor mínimo, entre otros. Por ejemplo, se puede almacenar las estadísticas descriptivas, en particular, el número de las observaciones del precio de los autos a través de la siguiente especificación[2]:

Ej. 2 ⟶
```
sum price
return list  // Para observar los elementos guardados con la ejecución de sum
scalar c =r(N)
```



Los comandos como sum simplemente analizan información sin estimar parámetros. Este tipo de comandos se conocen como r-class. Para ver la lista de escalares que se pueden guardar después de ejecutar un comando de tipo r-class se ejecuta el comando return list, como se muestra en el ejemplo 3. Los valores que se pueden almacenar como escalares luego de ejecutar un comando se encuentran al final de su documentación. Para guardar estos valores como escalares se utiliza la función r(), como se muestra a continuación:

Ej. 3 ⟶
```
sum mpg
return list
scalar media =r(mean)
scalar media =r(max)-r(min)
```

También es posible guardar el resultado de la ejecución de comandos para hacer estimaciones en escalares. Por ejemplo, el uso de escalares permite guardar el $R^2$ y otra información de una

[2] Utilizamos la base de datos de referencia de *Stata* sobre automóviles, puede acceder a ella a través de la siguiente sintaxis:

⟶ sysuse *auto*, clear

regresión. Para esto, suponga que se quiere explorar la relación del precio de un automóvil usado con otras características. El comando por ejecutar sería:

Ej. 4 ⟶ reg *price mpg rep78 headroom trunk weight length turn displacement gear_ratio foreign*

A este tipo de comandos, como reg, se conocen como e-class y para ver la lista de escalares que se pueden obtener de ellos se ejecuta el comando ereturn list. Para guardar el $R^2$, el número de observaciones o la suma de los residuales al cuadrado, se pueden ejecutar los siguientes comandos:

Ej. 5 ⟶
```
ereturn list // Para observar los elementos guardados con la ejecución de reg.
scalar a =e(r2)
scalar b =e(N)
scalar c =e(mss)
```

## 2. Matrices

### 2.1. Definición de vectores y matrices

*Stata* utiliza dos sintaxis diferentes en su programación para el uso de matrices. El primero de ellos se relaciona con los comandos que inician con el prefijo matrix. El segundo es a través del programa *Mata*, que tiene su propio lenguaje de programación bastante similar al utilizado en *Matlab,* que es de más bajo nivel y sobre el cual se construyen los comandos que hemos estudiado hasta ahora.



Las matrices se pueden crear directamente o se pueden utilizar para que almacenen resultados de estimaciones o de análisis de datos. Para crear matrices manualmente se utiliza el comando matrix define. Por ejemplo:

Ej. 6 ⟶
```
matrix define H= (1,0,4,5\8,0,3,7)
matrix list H
```

Las comas (,) separan columnas y la barra diagonal inversa (\) separa filas. Los nombres de las filas y las columnas pueden ser cambiados mediante el comando rownames o colnames, respectivamente:

Ej. 7 ⟶
```
matrix rownames H= Fila1 Fila 2
matlist H
```

Ahora, suponga que se busca almacenar los coeficientes estimados de una regresión por Mínimos Cuadrados Ordinarios (MCO). Al ejecutar el comando ereturn list después de la regresión se

observa que la matriz de coeficientes queda almacenada como e(b). Para guardar la matriz se utiliza el comando del ejemplo 6 abreviado:

Ej. 8 ⟶ matrix *coeficientes*=e(b)

Las formas abreviadas de los comandos pueden consultarse en su *help file*.

### 2.1. Obtener escalares a partir de matrices

*Stata* permite definir un escalar a partir de un elemento de una matriz. Para esto, es importante especificar la fila $i$ y la columna $j$. Por ejemplo, para obtener el coeficiente estimado para el peso del auto se debe especificar la fila 1 y la columna 5, así:

Ej. 9 ⟶ scalar *coef_peso*=coeficientes[1,5]

### 2.2. Operaciones básicas entre matrices

En *Stata* también se pueden hacer operaciones entre matrices como sumar, dividir, multiplicar, producto Kronecker, división por escalar, trasponer, entre otras.

Tabla 1. Operaciones entre matrices

| Operación | Resultado |
|---|---|
| B\C | Adjunta las filas de C |
| B , C | Adjunta las columnas de C |
| B + C | Suma de matrices |
| B - C | Resta de matrices |
| B * C | Multiplicación de matrices |
| B / C | División de matriz por un escalar |
| B:/C | División elemento por elemento |
| B # C | Producto kronecker |
| B' | Traspone matrices |



### 2.3. mkmat & svmat

Uno de los grandes beneficios del uso de matrices en *Stata* es la transformación que puede realizarse entre éstas y un grupo de variables. El comando mkmat permite transformar variables en matrices. La sintaxis básica del comando es:

mkmat varlist [if] [in] [, matrix(matname) nomissing]

Donde *varlist* hace referencia al conjunto de variables que se desea transferir en una matriz, la opción [, matrix(name)] permite definir el nombre para la matriz. La opción [, nomissing] es para eliminar de la nueva matriz los missing values de la base de datos.

Análogamente, el comando svmat permite realizar la transformación contraria, pasar de una matriz a una base de datos de variables. La sintaxis básica del comando es:

svmat [type] A [, names(col|eqcol|matcol|*string*)]

Donde la opción [type] indica el tipo de datos de las nuevas variables, *A* es el nombre de una matriz existente, [, names] indica los nombres que deben colocarse a las nuevas variables. En particular [, names(col)] indica a *Stata* quedarse con el nombre de las columnas de la matriz y [, names(*string*)] asigna nombres a las matrices que empiezan con el *string* especificado. Si el *string* es "inc", las variables serán nombradas: "inc1" "inc2" "inc3"… "inc*n*".

Para observar la aplicación de estos comandos se realiza una regresión múltiple por MCO usando matrices. La ecuación por estimar es:

$$price_i = \beta_0 + \beta_1 mpg_i + \beta_2 trunk_i + \beta_3 weight_i + \beta_4 length_i + \varepsilon_i$$



Se debe construir la matriz con los datos de las variables explicativas. Primero, se debe generar una variable igual a 1 para todas las observaciones que corresponda al vector de la constante ($\beta_0$). Posteriormente se generarán dos matrices, una para la variable dependiente, denotada Y, y otra para las variables independientes que se denota X. Para esto se ejecutan las siguientes líneas de programación:

Ej. 10 ⟶
```
gen constant=1
mkmat price, matrix(Y)
mkmat constant mpg trunk weight length, matrix(X)
```

En la creación de la matriz X, se colocan los nombres de las variables en el orden en que van a ser introducidas en la operación matricial. Recuerde que el estimador de MCO se expresa en forma matricial de la siguiente manera:

$$\beta = (X'X)^{-1}(X'Y)$$

Para generar la matriz *beta* (B) se deben realizar las operaciones matriciales necesarias, esto es, transponer, invertir y multiplicar. Las siguientes líneas de programación realizan el proceso para cada operación matricial:

Ej. 11 ⟶

```
matrix Xt=X'
matrix XtX=Xt*X
matrix XtX_inv=inv(Xt*X)
matrix XtY=Xt*Y
matrix B=(XtX_inv)*(XtY)
matrix list B
```

Generamos una matriz de una columna y como se puede observar el resultado es el mismo que la regresión por MCO usando el comando directamente. Finalmente, se busca que la matriz B sea una base de datos que guarde todos los coeficientes. Para traer la matriz de coeficientes como una base de datos se usa el comando svmat como se muestra en el ejemplo 12.

Ej. 12 ⟶

```
clear
matrix Beta=B'
svmat B, names(col)
```



Como resultado, ahora se tiene una base de datos con la estimación de los coeficientes.