

# Clase 4:

## Inicio básico y formas de organizar información

The logo for Stata, featuring the word "STATA" in a bold, blue, sans-serif font. The letters are slightly stylized, with the 'S' and 'A' having unique shapes.The logo for Python, featuring the word "python" in a gray, lowercase, sans-serif font, preceded by the Python logo icon which is a blue and yellow snake.

# Contenido

## 1. Inicio básico

- a. Interfaz
- b. Comentarios
- c. Directorio de trabajo
- d. Atajos de teclado en el editor
- e. Documentación

## 2. Formas de organizar información

- a. Variables
- b. Escalares y matrices
- c. Listas
- d. Diccionarios
- e. Dataframes

# 1. Inicio básico

The logo for Stata, featuring the word "STATA" in a bold, blue, sans-serif font. The letters are slightly italicized and have a modern, clean design.

## a. Interfaz de usuario

Barra de menú

Herramientas  
de acceso  
rápido

Ventana de  
comando

Ventana de  
resultados

Ventana de  
historial

Ventana del  
editor

Ventana de  
elementos

Visor de  
resultados

## b. Comentarios

Stata

R

Python

- Son útiles para agregar notas en el código
- Son una buena práctica de programación, ayudan a organizar el código

# Comentarios – Stata

```
1  * Colocar títulos y dar estructura al código
2
3  /* Realizar notas o explicaciones.
4   Se pueden hacer en varias líneas del do-file.
5   Hasta que se use el simbolo de final de comentario */
6
7  display 2+2 // Comentar líneas de código directamente
```

- Existen tres formas de crear comentarios
- " \* " y "/" son los símbolos relacionados con los comentarios

# Comentarios – R

```
1 # Los comentarios en R son iguales que en Python!  
2 # al incluir un "#" al principio de la línea, R la ignora para ejecutar
```

- Se crean antecediendo el símbolo #

# Comentarios – Python

```
#Todo lo que se encuentre después de un # no se ejecuta  
#como esta línea
```

- Se crean antecediendo el símbolo #



## c. Directorio de trabajo

Stata

R

Python

- Es la carpeta raíz donde estaremos trabajando, de allí importaremos y exportaremos los archivos necesarios
- Es útil definirlo al inicio del script

# Directorio de trabajo – Stata

Comando

```
cd "C:\Users\VEW0102\Curso Programación para datos EdContinua\Clases"
```

- Con el comando *cd* podemos fijar y conocer el directorio
- Entre comillas definimos la carpeta del ordenador

# Directorio de trabajo – R

```
#Set directory  
setwd("C:\\Users\\VEW0102\\Curso Programación para datos EdContinua\\Clases")
```

- Con el comando *getwd()* es posible conocer el directorio actual
- Con *setwd()* se fija el directorio

# Directorio de trabajo – Python

```
In [1]: pwd
```

```
Out[1]: 'C:\\Users\\VEW0102\\Universidad de los Andes\\'
```

```
%cd "C:\\Users\\VEW0102\\OneDrive - Universidad de los Andes\\Curso EdCo"
```

- El comando *pwd* nos permite conocer la ubicación actual del directorio
- Con *%cd* es posible cambiar el directorio

## d. Atajos de teclado del editor

**STATA**



python

	Stata		R		Python	
<i>Tarea</i>	<i>Windows</i> ⓘ	<i>Mac OS</i> ⓘ	<i>Windows</i> ⓘ	<i>Mac OS</i> ⓘ	<i>Windows</i> ⓘ	<i>Mac OS</i> ⓘ
Seleccionar línea	<i>Ctrl + N</i>	⌘ + <i>L</i>	<i>Ctrl + ⬆ + ←</i>	⌘ + ⬆ + ←	<i>Ctrl + ⬆ + ←</i>	⌘ + ⬆ + ←
Ejecutar órdenes	<i>Ctrl + D</i>	⌘ + ⬆ + <i>D</i>	<i>Ctrl + Enter</i>	⌘ + <i>Enter</i>	<i>Ctrl + Enter</i>	⌘ + <i>Enter</i>
Crear nuevo script	<i>Ctrl + N</i>	⌘ + <i>N</i>	<i>Ctrl + ⬆ + N</i>	⌘ + ⬆ + <i>N</i>	—	—
Comentar líneas	—	—	<i>Ctrl + ⬆ + C</i>	⌘ + ⬆ + <i>C</i>	<i>Ctrl + /</i>	⌘ + <i>/</i>
Guardar progreso	<i>Ctrl + S</i>	⌘ + <i>S</i>	<i>Ctrl + S</i>	⌘ + <i>S</i>	<i>Ctrl + S</i>	⌘ + <i>S</i>
Buscar	<i>Ctrl + F</i>	⌘ + <i>F</i>	<i>Ctrl + F</i>	⌘ + <i>F</i>	<i>Esc + F</i>	<i>Esc + F</i>
Reemplazar	<i>Ctrl + H</i>	⌘ + <i>Alt + F</i>	<i>Ctrl + F</i>	⌘ + <i>F</i>	<i>Esc + F</i>	<i>Esc + F</i>

**Nota:** Las funciones de copiar, cortar, pegar, deshacer y rehacer son las de cualquier procesador de texto. En R Studio, la combinación *Alt + ⬆ + K* permite ver todos los atajos de teclado disponibles. Con *Esc + P* se obtiene algo similar en Jupyter.

## d. Documentación

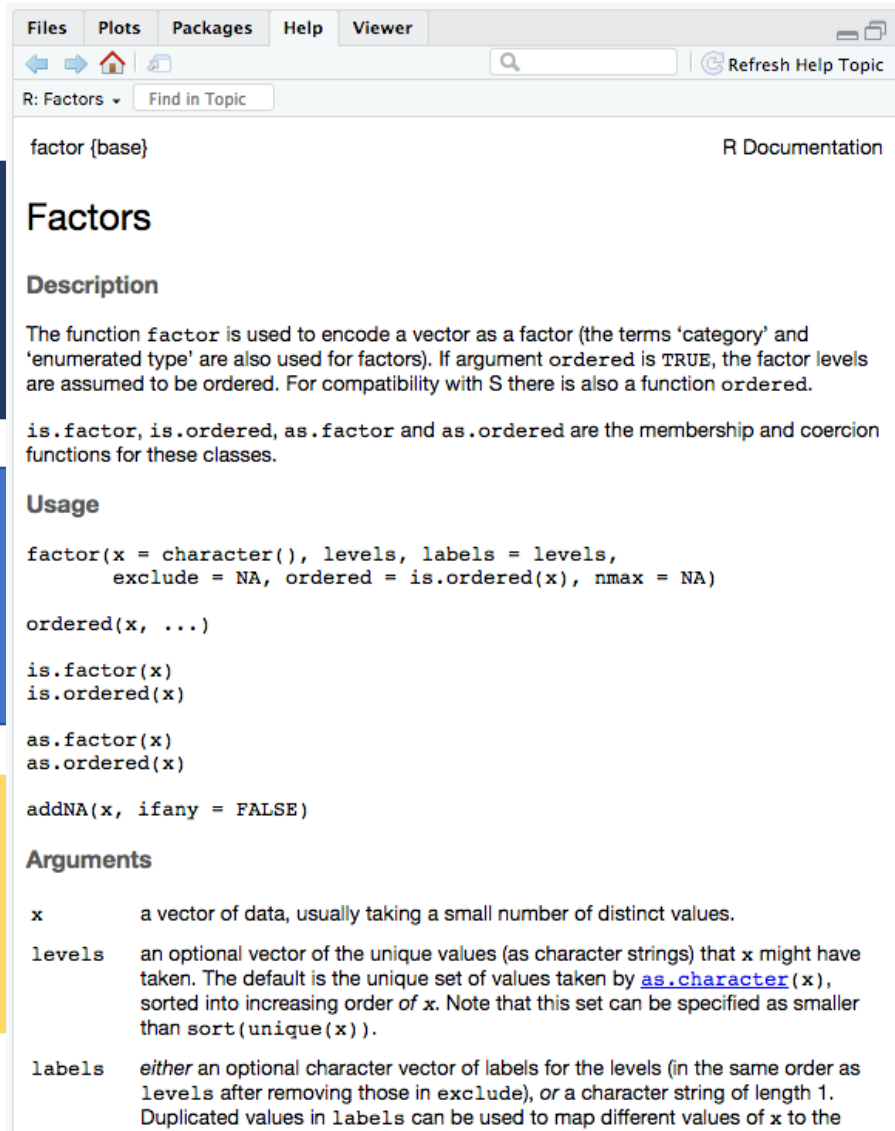
Stata

R

Python

- Descripción
- Uso (sintaxis)
- Argumentos
- Detalles
- Valor
- Advertencias
- Notas
- Referencias
- Ver también
- Ejemplos

# d. Documentación



The screenshot shows the R Documentation window for the 'factor' function. The window has a menu bar with 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. Below the menu bar is a search bar and a 'Refresh Help Topic' button. The main content area is titled 'factor {base}' and 'R Documentation'. It includes sections for 'Factors', 'Description', 'Usage', and 'Arguments'. The 'Description' section explains that the 'factor' function is used to encode a vector as a factor. The 'Usage' section lists the functions 'factor', 'ordered', 'is.factor', 'is.ordered', 'as.factor', 'as.ordered', and 'addNA'. The 'Arguments' section lists the arguments 'x', 'levels', and 'labels' with their descriptions.

**factor {base}** R Documentation

## Factors

### Description

The function `factor` is used to encode a vector as a factor (the terms 'category' and 'enumerated type' are also used for factors). If argument `ordered` is `TRUE`, the factor levels are assumed to be ordered. For compatibility with S there is also a function `ordered`.

`is.factor`, `is.ordered`, `as.factor` and `as.ordered` are the membership and coercion functions for these classes.

### Usage

```
factor(x = character(), levels, labels = levels,
       exclude = NA, ordered = is.ordered(x), nmax = NA)

ordered(x, ...)

is.factor(x)
is.ordered(x)

as.factor(x)
as.ordered(x)

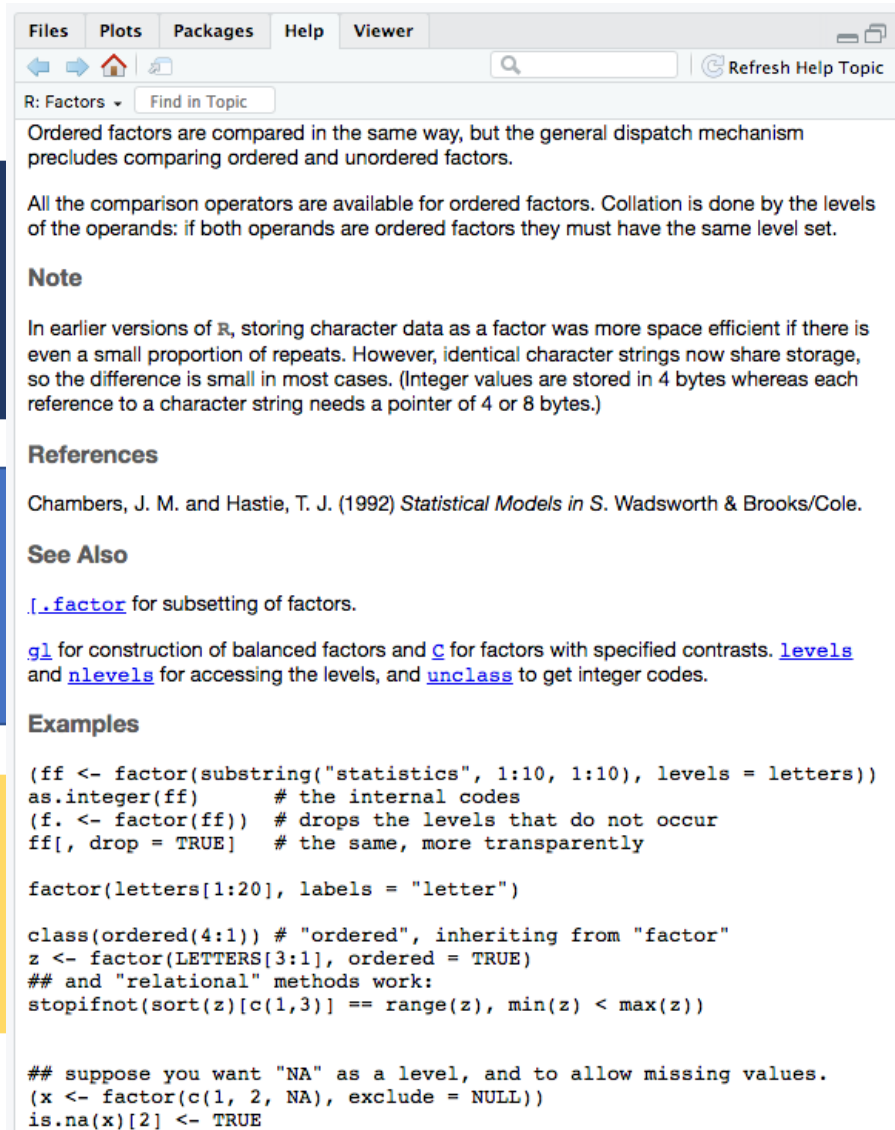
addNA(x, ifany = FALSE)
```

### Arguments

<code>x</code>	a vector of data, usually taking a small number of distinct values.
<code>levels</code>	an optional vector of the unique values (as character strings) that <code>x</code> might have taken. The default is the unique set of values taken by <code>as.character(x)</code> , sorted into increasing order of <code>x</code> . Note that this set can be specified as smaller than <code>sort(unique(x))</code> .
<code>labels</code>	<i>either</i> an optional character vector of labels for the levels (in the same order as levels after removing those in <code>exclude</code> ), <i>or</i> a character string of length 1. Duplicated values in <code>labels</code> can be used to map different values of <code>x</code> to the

- Descripción
- Uso (sintaxis)
- Argumentos
- Detalles
- Valor
- Advertencias
- Notas
- Referencias
- Ver también
- Ejemplos

# d. Documentación



Files Plots Packages Help Viewer

R: Factors Find in Topic

Ordered factors are compared in the same way, but the general dispatch mechanism precludes comparing ordered and unordered factors.

All the comparison operators are available for ordered factors. Collation is done by the levels of the operands: if both operands are ordered factors they must have the same level set.

**Note**

In earlier versions of R, storing character data as a factor was more space efficient if there is even a small proportion of repeats. However, identical character strings now share storage, so the difference is small in most cases. (Integer values are stored in 4 bytes whereas each reference to a character string needs a pointer of 4 or 8 bytes.)

**References**

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S*. Wadsworth & Brooks/Cole.

**See Also**

[\[.factor\]](#) for subsetting of factors.

[gl](#) for construction of balanced factors and [c](#) for factors with specified contrasts. [levels](#) and [nlevels](#) for accessing the levels, and [unclass](#) to get integer codes.

**Examples**

```
(ff <- factor(substring("statistics", 1:10, 1:10), levels = letters))
as.integer(ff)           # the internal codes
(f. <- factor(ff))       # drops the levels that do not occur
ff[, drop = TRUE]        # the same, more transparently

factor(letters[1:20], labels = "letter")

class(ordered(4:1)) # "ordered", inheriting from "factor"
z <- factor(LETTERS[3:1], ordered = TRUE)
## and "relational" methods work:
stopifnot(sort(z)[c(1,3)] == range(z), min(z) < max(z))

## suppose you want "NA" as a level, and to allow missing values.
(x <- factor(c(1, 2, NA), exclude = NULL))
is.na(x)[2] <- TRUE
```

- Descripción
- Uso (sintaxis)
- Argumentos
- Detalles
- Valor
- Advertencias
- Notas
- Referencias
- Ver también
- Ejemplos



# Documentación – Stata

Comando

**help regress**

- Se debe anteponer *help* al comando deseado
- Es posible pedir ayuda con la inicial *h*

# Documentación – R

```
6 # Para consultar la documentación usamos el signo de pregunta "?"  
7 ? factor
```

- Se debe anteponer el ? para consultar la ayuda

# Documentación – Python

```
help(print)
```

Help on built-in function print in module builtins:

```
print(...)
```

```
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.

sep: string inserted between values, default a space.

end: string appended after the last value, default a newline.

flush: whether to forcibly flush the stream.

- Se debe incluir dentro de paréntesis el comando deseado

## 2. Formas de organizar información

The logo for Stata, featuring the word "STATA" in a bold, blue, sans-serif font. The letters are slightly italicized and have a modern, clean design.The logo for Python, featuring the word "python" in a gray, lowercase, sans-serif font, preceded by the Python logo icon which consists of two interlocking snakes, one blue and one yellow.

# Tipos de datos

## Números

- Enteros
- Decimales
- Complejos

## Cadenas de texto (String)

- Con comillas simples
- Con comillas dobles

## Booleanos

- Lenguaje más básico de programación, 1 (*True*) y 0 (*False*)

# Formas de almacenamiento de información

Stata

R

Python

- a. Variables
- b. Matrices
- c. Listas
- d. Diccionarios
- e. Dataframes

## a. Variables

Stata

R

Python

- Son útiles para guardar información específica: nombres, valores, etc.
- Son elementos mutables pero no en Stata

# Variables – Stata

Comando

```
generate var=5
```

- Para crear variables usamos el comando *generate*



# Variables – R

```
11 edadDeMiHermana <- 17
12 nombreDelCurso = "Herramientas de programación"
```

```
>
> edadDeMiHermana
[1] 17
> nombreDelCurso
[1] "Herramientas de programación"
>
```

- Hay dos formas de darle un nombre a un valor como una variable

# Variables – Python

```
a = 5  
print(a)
```

```
5
```

- Se crean mediante el símbolo =

## b. Matrices

Stata

R

Python

- Almacenan información en más de una dimensión
- Son elementos mutables

# Matrices – Stata – Escalares

Ambiente  
base

```
. scalar a=5  
. display a  
5
```

Ambiente  
MATA

```
: a=5  
: a  
5
```

- Son matrices de orden 1

# Matrices – Stata

## Ambiente base

```
. matrix define lista=[2,4,6,8,10]

. matrix list lista

lista[1,5]
      c1  c2  c3  c4  c5
r1      2   4   6   8  10
```

## Ambiente MATA

```
: lista=(2,4,6,8,10)

: lista
      1   2   3   4   5
1  [ 2   4   6   8  10 ]
```

- Permiten almacenar números en un arreglo con filas y columnas
- Según el ambiente se pueden crear con un comando de forma directa

# Matrices – R

```
> matrix(1, 3, 3)
      [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
```

```
> A = matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3)
> A
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- El comando *matrix* permite crear un arreglo de cualquier dimensión
- También es posible usar la sintaxis *c(e1, e2, e3)* para crear vectores

# Matrices – Python

```
# Load library
import numpy as np

# Create a vector as a row
vector_row = np.array([1, 2, 3])

# Create a vector as a column
vector_column = np.array([[1],
                           [2],
                           [3]])
```

```
# Load library
import numpy as np

# Create a matrix
matrix = np.array([[1, 2],
                   [1, 2],
                   [1, 2]])
```

- Es necesario importar la biblioteca de *numpy*

## c. Listas

Stata

R

Python

- Permiten almacenar cadenas de texto y números en un solo lugar
- Son elementos mutables



# Listas – R

```
> my_list <- list(2,3)
> my_list
[[1]]
[1] 2

[[2]]
[1] 3
```

- Permiten almacenar diferentes tipos de datos
- Es posible acceder a cada elemento de la lista mediante el índice
- Existen elementos llamados tuplas que no son mutables

# Listas – Python

```
lista = [2,4,6,8,10]  
print(lista)
```

- Permiten almacenar diferentes tipos de datos
- Es posible acceder a cada elemento de la lista mediante el índice
- Existen elementos llamados tuplas que no son mutables

## d. Diccionarios

Stata

R

Python

- Facilitan el acceso a cada posición mediante el uso de claves
- Se crean usando llaves

# Diccionarios – R

```
> l <- list(a = 1,b = "foo",c = 1:5)
> l
$a
[1] 1

$b
[1] "foo"

$c
[1] 1 2 3 4 5
```

- No existen diccionarios en R, pero las listas trabajan de la misma manera

# Diccionarios – Python

```
diccionario = {'1':"Es", '2':"un", '3':"día"}  
print(diccionario)
```

```
{'1': 'Es', '2': 'un', '3': 'día'}
```

```
#Agregando un elemento  
diccionario['4']='soleado'  
print(diccionario)
```

```
#Eliminando elementos  
del(diccionario['1'])  
print(diccionario)
```

- No permite llaves duplicadas
- Almacena cualquier tipo de datos

## e. Dataframes

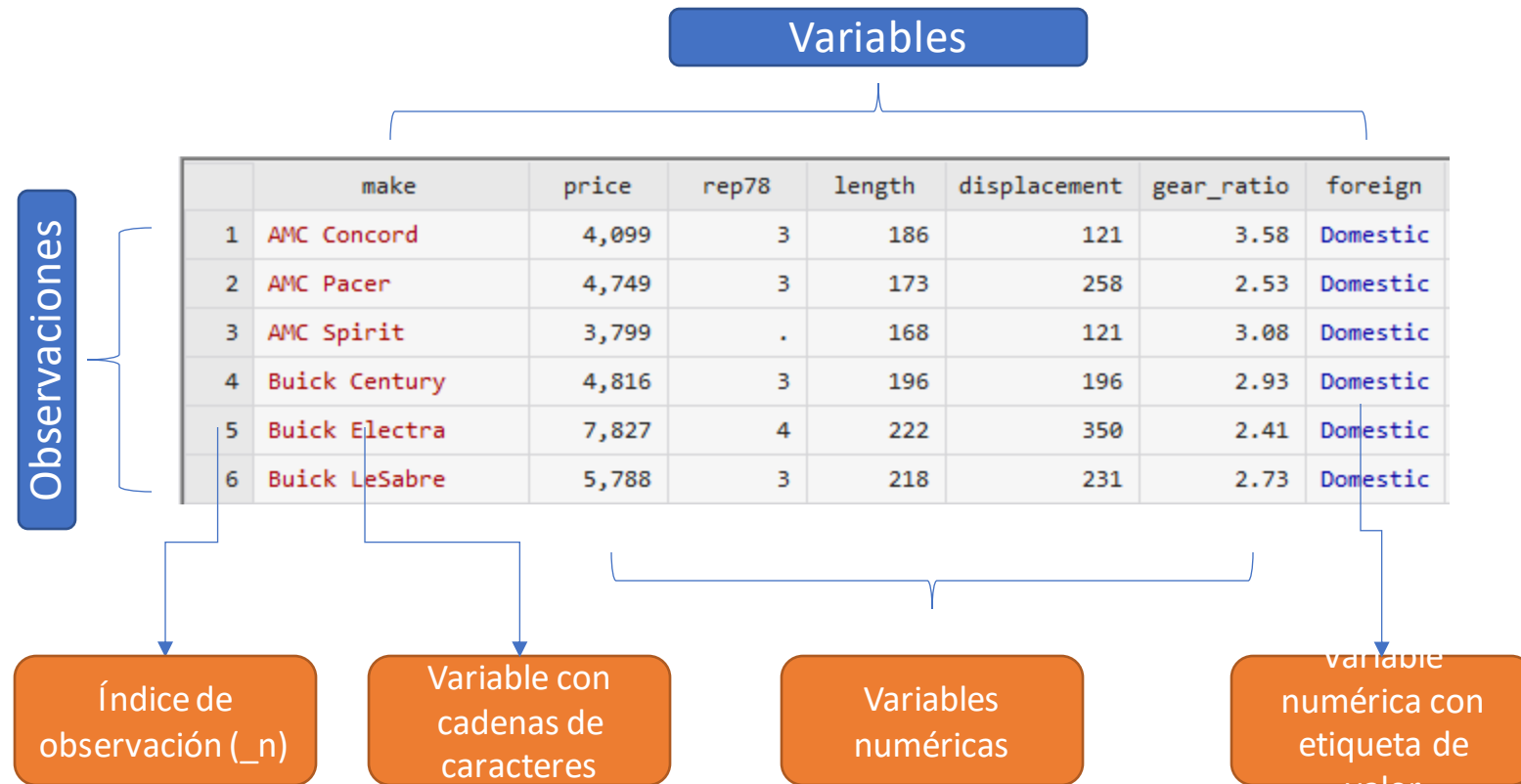
Stata

R

Python

- Son las formas más usadas en el análisis de datos

# Dataframes – Stata



# Dataframes – R

	wage	hours	IQ	educ	exper	tenure	age	married	black	urban	brthord	meduc	feduc	lwage
1	769	40	93	12	11	2	31	1	0	1	2	8	8	6.645091
2	808	50	119	18	11	16	37	1	0	1	.	14	14	6.694562
3	825	40	108	14	11	9	33	1	0	1	2	14	14	6.715384
4	650	40	96	12	13	7	32	1	0	1	3	12	12	6.476973
5	562	40	74	11	14	5	34	1	0	1	6	6	11	6.331502
6	1400	40	116	16	14	2	35	1	1	1	2	8	.	7.244227
7	600	40	91	10	13	0	30	0	0	1	2	8	8	6.396930
8	1081	40	114	18	8	14	38	1	0	1	3	8	.	6.985642
9	1154	45	111	15	13	1	36	1	0	0	3	14	5	7.050990
10	1000	40	95	12	16	16	36	1	0	1	1	12	11	6.907755
11	930	43	132	18	8	13	38	1	0	0	1	13	14	6.835185
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Showing 1 to 12 of 935 entries, 14 total columns



# Dataframes – Python

```
import pandas as pd
```

```
df= pd.read_csv("C:/Users/VEW0102/OneDrive - Universidad de los Andes/9
```

```
print(df)
```

	wage	hours	IQ	educ	exper	tenure	age	married	black	urban
0	769	40	93	12	11	2	31	1	0	1
1	808	50	119	18	11	16	37	1	0	1
2	825	40	108	14	11	9	33	1	0	1
3	650	40	96	12	13	7	32	1	0	1
4	562	40	74	11	14	5	34	1	0	1
..	...	...	...	...	...	...	...	...	...	...
930	520	40	79	16	6	1	30	1	1	0
931	1202	40	102	13	10	3	31	1	0	1
932	538	45	77	12	12	10	28	1	1	0
933	873	44	109	12	12	12	28	1	0	0
934	1000	40	107	12	17	18	35	1	0	0