# Funciones y matrices

Herramientas de programación para el análisis de datos Clase 6

# ¿Cómo se sienten hasta ahora?

- Hemos comenzado a trabajar código
- La clase pasada incluía bastante tema
- Les vamos a enviar una encuesta

# ¿Han utilizado matrices alguna vez?

Tipo matemáticas, no muy avanzado pero con un par de operaciones

# Reto: quieren calcular el promedio de una secuencia de números

Sentados frente a un computador ¿cómo lo hacen? (en algún lenguaje cualquiera)

- Tomar valores y sumarlos, después dividirlos entre el número de valores
- Invocar la función de promedio y decirle qué valores utilizar

## Hoy

- Funciones
- Matrices
- Ejercicio para prototipar código complejo

### Funciones pequeño ejemplo: invocar el promedio

```
// Tenemos una variable, e invocamos descriptivas
sum nombreDeLaVariable
```

```
valores <- c(1, 2, 3, 4)
promedio <- mean(valores)</pre>
```

```
import statistics
valores = [1, 2, 3, 4]
promedio = statistics.mean(valores)
```

- Nos sirven para ejecutar tareas específicas.
- Son útiles cuando esas tareas se repiten.
- Para usarlas tenemos que darles información en cada situación (parámetros).
- Para usar una función buscamos en internet la documentación

## Un pequeño foco en Stata



CASO 1 (lo común)

Queremos saber y usar la media. Entonces usamos **sum**.

## Un pequeño foco en Stata

valores	nuevaCol
334	3726.25 ue creen que
3245	va a ocurrir?
5643	3726.25 (Un minuto para
5683	pensar) 3726.25

egen nuevaCol = mean(valores)

г у споп

CASO 1 (lo común)

Queremos saber y usar la media. Entonces usamos **sum**.

CASO 2 (uno avanzado)

Queremos crear una columna nueva con el valor de la media. Entonces usamos **egen** 

## ¡Ejercicio!

```
valores:
5, 7, 9, 2, 4
```

- Escoja un lenguaje cualquiera.
- Arme el siguiente arreglo/columna de valores
- Busque la documentación en ese lenguaje para

Saber el valor máximo

## ¡Ejercicio!



- Escoja un lenguaje cualquiera.
- Arme el siguiente arreglo/columna de valores
- Busque la documentación en ese lenguaje para

Saber el valor máximo

# ¿Cómo crear nuestras propias funciones?

Digamos que quieren calcular un indicador que ustedes se inventaron

## Crear nuestras propias funciones

```
// En Stata: normalmente no creamos funciones
nuevas, todo queda en el do-file pero se puede
hacer agregando archivos ado (nuevos programas) al
sistema
```

```
nombre_de_la_fun <- function(arg_1, arg_2, ...) {
   return(3)
}</pre>
```

```
def nombre_de_la_fun(arg_1, arg_2, ...):
    return 3
```

#### Hay que definir:

- El nombre con el que se va a invocar
- Los parámetros
- El proceso
- Lo que retorna

Nombren cada uno en el caso de la función media

### Detalle de R

```
En Stata: normalmente no creamos funciones
cuentame <- function(valor1){</pre>
  valor1 + 1
# ¿Qué valor retorna?
cuentame(2)
                             Retorna 3
```

Cuando no especificamos qué valor retornar, R retorna la última expresión evaluada.

## Crear nuestras propias funciones

```
// En Stata: normalmente no creamos funciones
nuevas, todo queda en el do-file pero se puede
hacer agregando archivos ado (nuevos programas) al
sistema
```

```
nombre_de_la_fun <- function(arg_1, arg_2, ...) {
    # Cuerpo de la función
    return(3)
}</pre>
```

```
def nombre_de_la_fun(arg_1, arg_2, ...):
    return 3
```

#### Hay que definir:

- El nombre con el que se va a invocar
- Los parámetros
- El proceso
- Lo que retorna

Nombren cada uno en el caso de la función media

## El caso de Python

```
// En Stata: normalmente no creamos funciones
nuevas, todo queda en el do-file nero se nuede
hac
   def unaFuncionMuyBuena(dato1, dato2):
non
           retornar = dato1+dato2
           return retornar
   # ¿Qué valor retorna?
   unaFuncionMuyBuena(2,3)
```

En Python, si no especificamos el valor de retorno, entonces retorna **None** 

## Algo muy i

# Un objeto en Python se define como class Persona:

## <u>si usamos objetos</u>

```
# ¡OJO! Esto es por si van a definir funciones DENTRO DE UN OBJETO:
hac c
      La función debe recibir un parámetro "self"
                                                                                       lla
     # Pero en general no hay que hacer eso, solo definimos:
                                                                                        para
non
                                                                                        e usaría
      def funcion(param1, param2):
                                                                                       ad()
                                                               <del>estoy preguntando su ed</del>ad
   marcus.dameTuEdad()
```

#### Síntesis

```
// En Stata: normalmente no creamos funciones
nuevas, todo queda en el do-file pero se puede
hacer agregando archivos ado (nuevos programas) al
sistema
nombre_de_la_fun <- function(arg_1, arg_2, ...) {</pre>
   # Cuerpo de la función
   return(3)
def nombre_de_la_fun(arg_1, arg_2, ...):
       return 3
```

## Ejercicio de dificultad intermedia (10 minutos)

```
En R: datos <- c(6,3,6,3,2)
En Python: datos = [6,3,6,3,2]
```

Defina una función en uno de los lenguajes que está utilizando que:

- Tome un vector de valores
- Recupere el primer valor
- Recupere el último valor de la lista
- Retorne la suma de esos dos

## Ejercicios activos

Respuesta: 8

La clave es utilizar la función

longitud:

En R: length(datos)

En Python: len(datos)

Veamos un par de ejemplos.

## Matrices

# Advertencia: hay dos formas de "arreglar"

¿Para qué se usan?

**Matrices** 

 $\begin{bmatrix} 2 & 5 & 5 & 1 \\ 2 & 4 & 9 & 3 \\ 2 & 3 & 3 & 1 \\ 6 & 7 & 6 & 8 \end{bmatrix}$ 

s: llámense base de datos, dataFrames, etc.

Columna 1	Columna 2	Columna 3	Columna 4
2	5	5	1
2	4	9	3
2	3	3	1

Ustedes han visto ya este un poco y lo exploraremos más en el módulo 4

### Detrás de bambalinas

• Cuando ustedes llaman funciones que calculan cosas con los datos:

#### una media

• Detrás de esos cálculos hay matrices

#### Al final del curso vamos a explorar "regresiones"

- Esto usa muuuchas matrices detrás de bambalinas
- Mucha de la estadística se construye usando Álgebra lineal

## Para qué las usamos en el mundo de los datos

- En stata: a veces hacemos cálculos y queremos irlos guardando organizadamente para luego anotarlos en un archivo.
- Si queremos implementar a mano funciones que ya existen (porque queremos cambiarles algo particular sobre su metodología).
- ¡Cuando **jugamos con matemáticas**! Por ejemplo queremos estudiar cómo se comporta una montañita descrita por una fórmula

# Juguemos con matrices

## Les presento a A, B y C

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

## Suma A + B (Alf en vivo)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\mathsf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## Resta A – C (Ejercicio)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC \left[ \begin{array}{c} 1 \cdot 1 + \\ \end{array} \right]$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC \left[ \begin{array}{c} 1 \cdot 1 + 2 \cdot 2 + \\ \end{array} \right]$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC \begin{bmatrix} 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 2 \\ AC \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC \begin{bmatrix} 1+2\cdot 2+3\cdot 2 \\ \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC \left[ \begin{array}{c} 1+4+3\cdot 2 \\ \end{array} \right]$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC \begin{bmatrix} 1+4+6 \\ 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC$$
 $11$ 
Ejercici

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC\begin{bmatrix} 1 \cdot 2 + 2 \cdot 1 + 3 \cdot 1 & 11 \\ \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

$$AC\begin{bmatrix} 7 & 11 \\ & & \end{bmatrix}$$

#### Nota

# A por B no es igual a B por A

En matrices, el orden importa

# Una matriz útil: la matriz transpuesta

Transponer:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix}$$

Cada filas se vuelve una columna:

$$A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$C^T = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 3 & 1 \end{bmatrix}$$

# Multiplicación de matriz por escalar

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 2 \cdot B = \begin{bmatrix} 2 \cdot 1 & 2 \cdot 0 & 2 \cdot 0 \\ 2 \cdot 0 & 2 \cdot 1 & 2 \cdot 0 \\ 2 \cdot 0 & 2 \cdot 0 & 2 \cdot 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

# Ahora sí matrices en código

Ya sabiendo esto es súper sencillo

# ¿Recuerdan los arreglos?

```
// Por ejemplo:
local datos 6 7 8 9
```

```
datos <- c(6,7,8,9)
datos[2] # nos muestra 7</pre>
```

```
datos = [6,7,8,9]
datos[2] # nos muestra 8
```

Para recuperar un valor de un arreglo, tenemos que comunicar un dato que indique la posición del valor que queremos recuperar.

En Stata no solemos hacer eso con macros.

## ¿Recuerdan los arreglos?

```
// Por ejemplo:
local datos 6 7 8 9
```

```
datos <- c(6,7,8,9)
datos[2] # nos muestra 7
```

```
datos = [6,7,8,9]
datos[2] # nos muestra 8
```

Para tomar un valor de una matriz tenemos que indicar **dos datos**: fila y columna

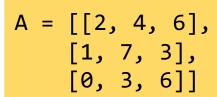
$$\begin{pmatrix} 2 & 4 & 6 \\ 1 & 7 & 3 \\ 0 & 3 & 6 \end{pmatrix}$$

#### Matrices

```
matrix A = (2,4,6\1,7,3\0,3,6)
```

Para crear esta matriz en cada lenguaje, también las definimos en términos de filas y columnas

```
arreglo1 <- c(2,4,6)
arreglo2 <- c(1,7,3)
arreglo3 <- c(0,3,6)
matrix <- cbind(arreglo1, arreglo2, arreglo3)</pre>
```



Note: R pega columnas Python y STATA pegan filas

 $\begin{array}{ccc}
 & 4 & 6 \\
 & 7 & 3 \\
 & 3 & 6
\end{array}$ 

#### Creando matrices en Stata desde variables

En Stata también podemos partir del editor y convertir columnas enteras en matriz

#### Matrices

```
matrix A = (2,4,6\1,7,3\0,3,6)
```

```
arreglo1 <- c(2,4,6)
arreglo2 <- c(1,7,3)
arreglo3 <- c(0,3,6)
matrix <- cbind(arreglo1, arreglo2, arreglo3)</pre>
```

```
A = [[2, 4, 6],
[1, 7, 3],
[0, 3, 6]]
```

Para crear esta matriz en cada lenguaje, las definimos en términos de filas y columnas

$$\begin{pmatrix} 2 & 4 & 6 \\ 1 & 7 & 3 \\ 0 & 3 & 6 \end{pmatrix}$$

#### Otras formas de crear matrices en R

```
A <- matrix(1:9, nrow = 3, ncol = 3)
  [[م رد ره]
```

Hay atajos para crear matrices en R cuando son especiales.

- Números secuenciales
- Ceros
- Identidad

En STATA y Python también

#### Matrices

```
matrix A = (2,4,6\1,7,3\0,3,6)
```

```
arreglo1 <- c(2,4,6)
arreglo2 <- c(1,7,3)
arreglo3 <- c(0,3,6)
matrix <- cbind(arreglo1, arreglo2, arreglo3)</pre>
```

```
A = [[2, 4, 6],
[1, 7, 3],
[0, 3, 6]]
```

Para crear esta matriz en cada lenguaje, las definimos en términos de filas y columnas

$$\begin{pmatrix} 2 & 4 & 6 \\ 1 & 7 & 3 \\ 0 & 3 & 6 \end{pmatrix}$$

# En Python también se pueden crear matrices con numPy

```
mat
   # Importamos numpy
   import numpy as np
   A = np.matrix('2 4 6; 1 7 3; 0 3 6')
```

Para estadística es más común usar matrices de la librería **numpy** 

# En Python también se pueden crear matrices con numPy

```
# Definida nativamente
   Anat = [[2, 4, 6],
        [1, 7, 3],
        [0, 3, 6]]
   # Definida mediante numpy
   import numpy as np
   Anum = np.matrix('2 4 6; 1 7 3; 0 3 6')
mat
             [40]: Anat
             [40]: [[2, 4, 6], [1, 7, 3], [0, 3, 6]]
             [41]:
                  Anum
             [41]: matrix([[2, 4, 6],
```

Para estadística es más común usar matrices de la librería **numpy** 

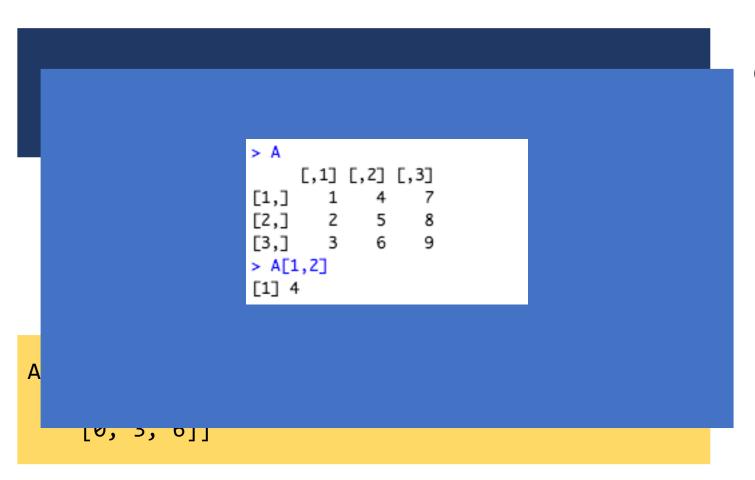
# En Python también se pueden crear matrices con numPy

```
# Definida nativamente
   Anat = [[2, 4, 6], [1, 7, 3],
        [0, 3, 6]]
    # Definida mediante numpy
    import numpy as np
    Anum = np.matrix('2 4 6; 1 7 3; 0 3 6')
mat
     elemento en fila 0 columna 1 (OJO CONTANDO DE 0)
                [38]: Anat[0][1]
                [38]: 4
                [29]: Anum.item((0, 1))
                [29]: 4
```

Para estadística es más común usar matrices de la librería **numpy** 

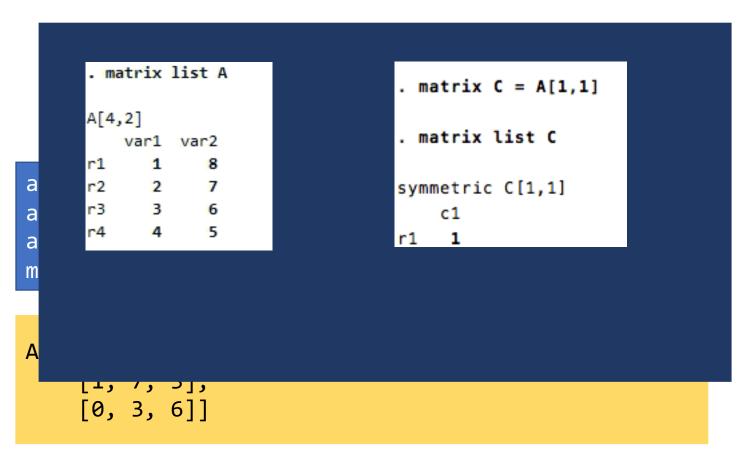
No importa si usan matrices nativas o de numPy, pero sean consistentes.

#### Tomar elementos de una matriz en R



Para seleccionar elementos de una matriz en R se usa la notación con corchetes

#### Tomar elementos de una matriz en STATA



Se pueden asignar partes específicas de una matriz: Recuede que STATA cuenta desde 1.

# Operaciones

# Suma

$$matrix C = A + B$$

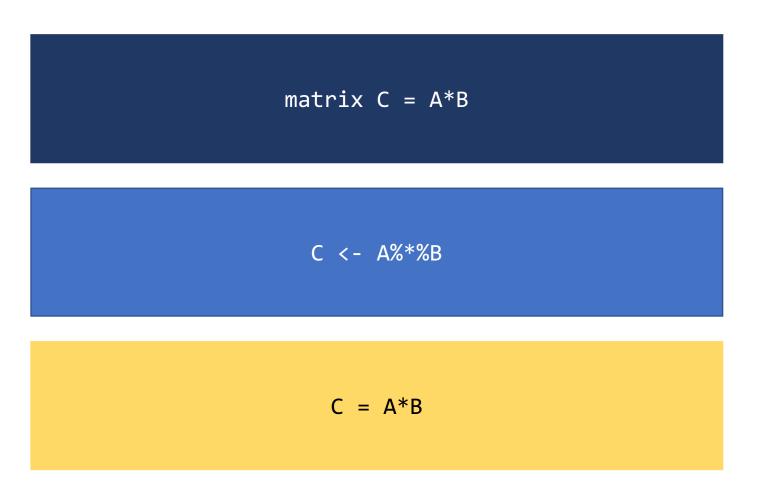
$$C = A + B$$

### Resta

matrix C = A - B

$$C = A - B$$

# Multiplicación



En R consideran que la multiplicación es más común es elemento a elemento, por eso la notación simple está reservada para "elemento por elemento"

# Multiplicación elemento por elemento

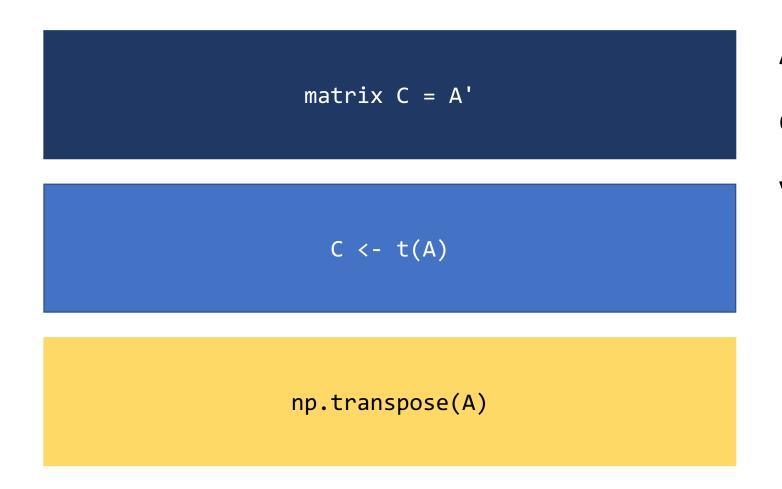
```
// No acepta (hay que programarla con loops)
```

C <- A\*B

C = np.multiply(A,B)

En python las librerías para matrices están programadas en numpy.

# Transposición: filas a columnas



Aquí sí, la notación de cada lenguaje es única y especial.

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vamos a crear una función en R o Python que tome una matriz:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- En la página está esta presentación para referencia
- Vamos a dar 2 minutos para cada prototipo parcial
- Alfredo va a ejecutar el prototipo parcial después: nadie se nos queda

- 1. Alf: Dibujar el paso a paso
- 2. Alf: Describir el paso a paso en voz alta (como lo haría un computador)

- 1. Alf: Dibujar el paso a paso
- 2. Alf: Describir el paso a paso en voz alta
- 3. Ustedes: Crear una matriz 3x3 llena de ceros
- 4. Ustedes: Crear una función que muestre cualquier cosa en pantalla y reciba un parámetro
- 5. Ustedes: hacer un loop que muestre en pantalla cada número de fila
- 6. Ustedes: hacer un loop dentro de ese loop que muestre cada número de columna, y que muestre en qué fila va y en qué columna
- 7. Ustedes: afuera de todo eso, cambiarle el valor a una celda de la matriz (cualquiera)
- 8. Ustedes: meter dentro del doble loop la asignación anterior