DigitalHouse > Coding School

DATA SCIENCE

MÓDULO 1

Numpy

NumPy

OBJETIVOS DE LA CLASE

1. **Manipular datos con Arrays de NumPy**
2. **Vectorización de operaciones y optimización**
3. **Acceder a los datos de forma eficiente**

NumPy es el paquete fundamental para la computación científica con Python.
Contiene, entre otras cosas:

- un poderoso elemento: los **arrays** de N-dimensiones (ND arrays)
- Funciones para realizar cálculos y operaciones matemáticas con arrays de forma muy eficiente (vectorización)
- Operaciones de álgebra lineal, generación de números aleatorios, entre otras funcionalidades de computación científica

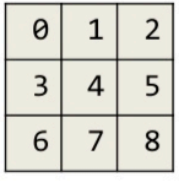
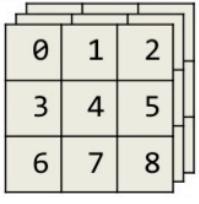
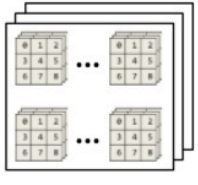
Esto permite a NumPy se integre de manera rápida y sin problemas con una amplia variedad de bases de datos.

Un arreglo es una **colección de items de datos**, llamados elementos, asociados a un único nombre de variable.

Diseñados para:

- Facilitar la programación y proveer buena performance.
- Almacenar y manipular grandes colecciones de datos.
- Simplificar la tarea de nombrar y referenciar ítems individuales en una colección de datos.
- Permitir manipular una colección entera de datos con una simple sentencia.

Un **arreglo** es una estructura de datos que permite agrupar varios valores bajo un único nombre. Los elementos de un arreglo son todos del mismo tipo (a diferencia de las listas de Python).

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3rd order Tensor)
N		ND Array

- Un vector es un arreglo unidimensional
- Una tabla o matriz es un arreglo bidimensional
- Por último un arreglo puede tener N dimensiones.

— **Creación**:

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
b = np.array([5, 4, 3, 2, 1])
```

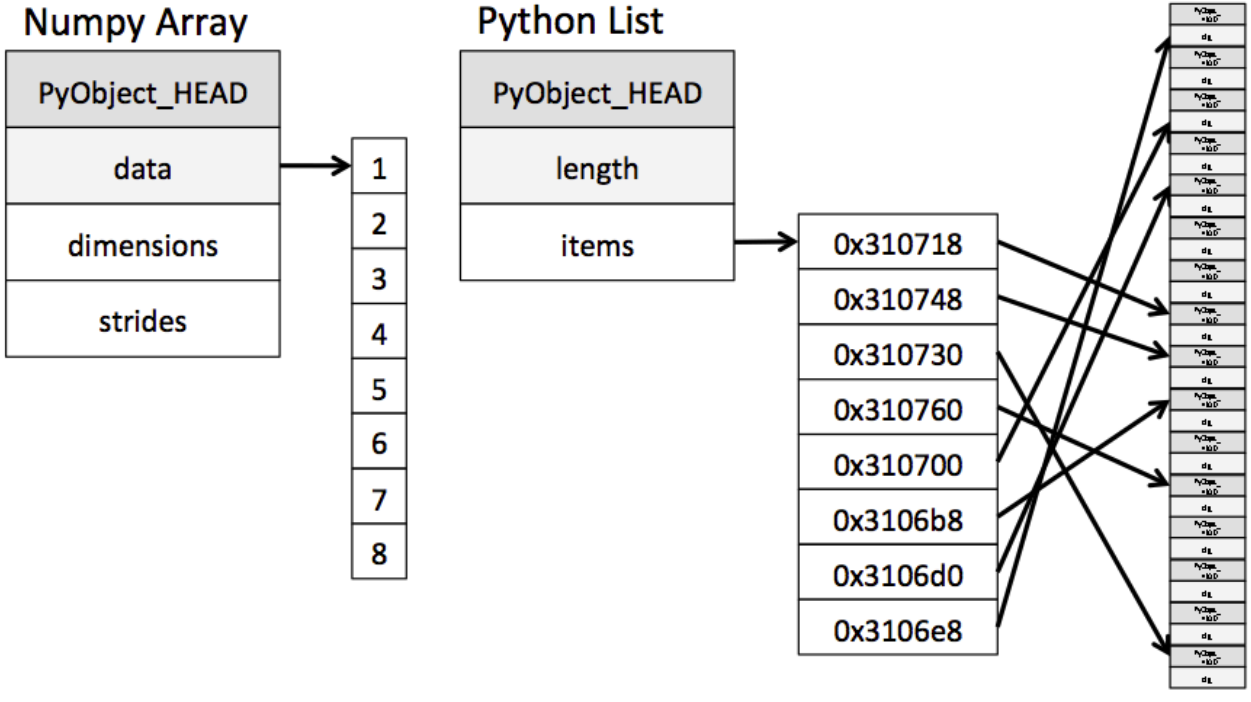
— **Acceso / Indexación**:

```
a[0]
```

Los arreglos se indexan desde 0 a len(a) -1:

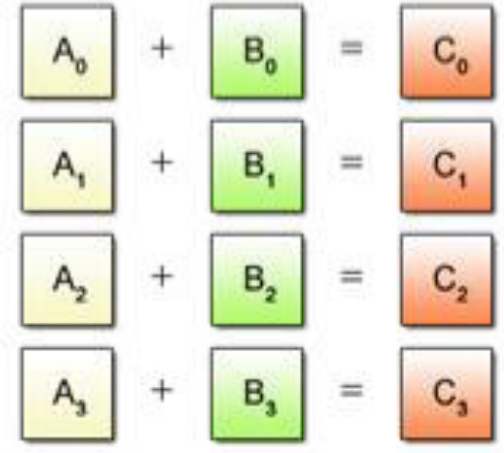
— a[0] devuelve el valor de la primera posición del arreglo
— a[len(a)-1] devuelve el valor de la última posición del arreglo

Los Arrays de Numpy son estructuras mucho más eficientes para operar sobre los datos que las listas de Python. Esto en parte se debe a que en un Numpy Array el tipo de datos es fijo.

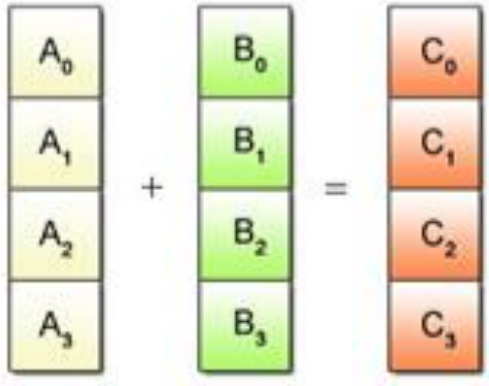


Las operaciones vectorizadas trabajan sobre los **datos como un bloque**. Por eso es necesario que los tipos sean **homogéneos** entre todos los elementos. En la operación vectorizada, no recorremos los elementos en orden.

Operación escalar



Operación vectorizada



Tipos de datos en Numpy:

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long; normally either int64 or int32)
intc	Identical to C int (normally int32 or int64)
intp	Integer used for indexing (same as C ssize_t; normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128.
complex64	Complex number, represented by two 32-bit floats
complex128	Complex number, represented by two 64-bit floats

Ejemplo de operaciones vectorizadas:

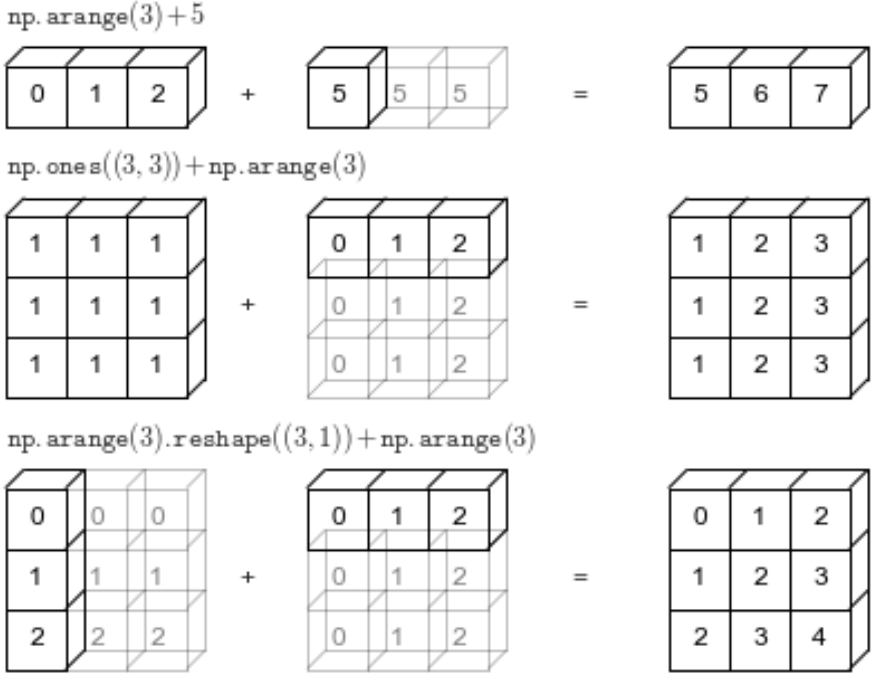
```
In [7]: x = np.arange(4)
        print("x     =", x)
        print("x + 5 =", x + 5)
        print("x - 5 =", x - 5)
        print("x * 2 =", x * 2)
        print("x / 2 =", x / 2)
        print("x // 2 =", x // 2)  # floor division
```

```
x     = [0 1 2 3]
x + 5 = [5 6 7 8]
x - 5 = [-5 -4 -3 -2]
x * 2 = [0 2 4 6]
x / 2 = [ 0.   0.5  1.   1.5]
x // 2 = [0 0 1 1]
```

Operator	Equivalent ufunc	Description
+	np.add	Addition (e.g., 1 + 1 = 2)
-	np.subtract	Subtraction (e.g., 3 - 2 = 1)
-	np.negative	Unary negation (e.g., -2)
*	np.multiply	Multiplication (e.g., 2 * 3 = 6)
/	np.divide	Division (e.g., 3 / 2 = 1.5)
//	np.floor_divide	Floor division (e.g., 3 // 2 = 1)
**	np.power	Exponentiation (e.g., 2 ** 3 = 8)
%	np.mod	Modulus/remainder (e.g., 9 % 4 = 1)

Numpy tiene un conjunto de reglas para aplicar operaciones miembro a miembro en Arrays de diferente tamaño.

Se **proyectan** los valores de los arrays para poder operar sobre los mismos.



El broadcasting en NumPy sigue un conjunto estricto de reglas para determinar la interacción entre las dos arrays:

Regla 1: si los dos arrays difieren en su número de dimensiones, la forma de la que tiene menos dimensiones se rellena con unos en su lado delantero (izquierdo).

Regla 2: si la forma de los dos arrays no coincide en alguna dimensión, el array con forma igual a 1 en esa dimensión se estira para que coincida con la otra.

Regla 3: si en alguna dimensión los tamaños son diferentes y ninguno es igual a 1, se genera un error.

Ejemplo 1

```
M = np.ones((2, 3))
a = np.arange(3)
```

```
M.shape = (2, 3)
a.shape = (3,)
```

(por regla 1)

```
M.shape -> (2, 3)
a.shape -> (1, 3)
```

(por regla 2)

```
M.shape -> (2, 3)
a.shape -> (2, 3)
```

```
M + a
```

```
array([[ 1.,  2.,  3.],
       [ 1.,  2.,  3.]])
```

Ejemplo 2

```
a = np.arange(3).reshape((3, 1))
b = np.arange(3)
```

```
a.shape = (3, 1)
b.shape = (3,)
```

(por regla 1)

```
a.shape -> (3, 1)
b.shape -> (1, 3)
```

(por regla 2)

```
a.shape -> (3, 3)
b.shape -> (3, 3)
```

```
a + b
```

```
array([[0, 1, 2],
       [1, 2, 3],
       [2, 3, 4]])
```

Ejemplo 3

```
M = np.ones((3, 2))
a = np.arange(3)
```

```
M.shape = (3, 2)
a.shape = (3,)
```

(por regla 1)

```
M.shape -> (3, 2)
a.shape -> (1, 3)
```

(por regla 2)

```
M.shape -> (3, 2)
a.shape -> (3, 3)
```

(por regla 3)

```
M + a
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-9e16e9f98da6> in <module>()
```

Llamamos “indexing” al proceso de acceder a los elementos de un array con algún criterio.
Existen tres tipos de indexing en Numpy:

- Slicing: cuando accedemos a los elementos con los parámetros *start,stop,step*.
  Por ejemplo my_array[0:5:-1]

- Fancy Indexing: Cuando creamos una lista de índices y la usamos para acceder a ciertos elementos del array: my_array[[3,5,7,8]]

- Boolean Indexing: Cuando creamos una “máscara booleana” (un array o lista de True y False) para acceder a ciertos elementos: my_array[my_array > 4]

Son conjuntos de recursos que se incorporan en un sistema para aprovechar su funcionalidad.

- **Reutilización entre distintos sistemas**
- **Estandarización**
- **Aprovechar desarrollos complejos que ya están hechos**
- **Trabajar en línea con la comunidad**

Algunas de las bibliotecas que vamos a usar en este curso

- **NumPy**
  - Permite trabajar de manera eficiente con operaciones matemáticas sobre arrays
- **Pandas**
  - Se usa para análisis y manipulación de datos como tablas (dataframes)
- **Matplotlib**
  - Es una librería de visualización, para gráficos y tableros
- **Seaborn**
  - Otra librería de visualización, basada en matplotlib

PRÁCTICA GUIADA

Utilizar los datasets volumen_ventas_producto_sucursal.csv y precio_producto.csv que indican las ventas en cantidad para cada producto en cada sucursal y el precio de cada producto respectivamente.

Ejercicios:

- Calcular la cantidad de unidades vendidas para cada sucursal
- Calcular la cantidad de unidades vendidas para cada producto
- ¿Cuál es la sucursal que más vendió?
- Calcular la facturación de cada sucursal

INTRODUCCIÓN ESTADÍSTICA

OBJETIVOS DE LA CLASE

1. **Repasar las medidas de tendencia central (media, mediana y moda)**
2. **Repasar cómo la media, mediana y moda son afectadas por la asimetría**
3. **Estudiar las medidas de correlación y covarianza.**
4. **Utilizar la librería numpy y scipy**
5. **Realizar un laboratorio integrador de lo que hemos estado haciendo hasta ahora**

- Existen dos campos en la estadística:
  - **Descriptiva**
  - **Inferencial**

- **Foco en estadística descriptiva**: describir, sumarizar y comprender los datos.

- **Medidas de Tendencia Central**: proveer información descriptiva sobre el valor numérico que es considerado el más usual para una variable cuantitativa:
  - **Media**
  - **Mediana**
  - **Moda**

- **Asimetría** en la distribución de datos. Efecto en la media, mediana y moda.

- **Medidas de Variabilidad:**
  - **Rango**
  - **Varianza**
  - **Desvío Estándar**
  - **Coeficiente de Variación**
- NumPy tiene funciones para calcular todas estas medidas, pero antes vamos a ir a los conceptos fundamentales.

La **media** se define de la siguiente manera:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Por ejemplo, para la muestra 8, 5 y -1, su media es:

$$\bar{x} = \frac{8 + 5 + (-1)}{3} = 4$$

La **mediana** puede pensarse de manera simple como el valor del ”medio” de una lista ordenada de datos (o el valor que separa la primera mitad y la segunda mitad de una distribución).
Para una lista ordenada la mediana es calculada de diferente manera dependiendo de la cantidad de elementos de la misma:

- **Impar:**

  [1, 2, 3, 5, **7**, 8, 9, 10, 15]
  #elementos: 9
  La mediana es el valor de la posición 5 (la posición del “medio”)
  Mediana = 7

- **Par:**

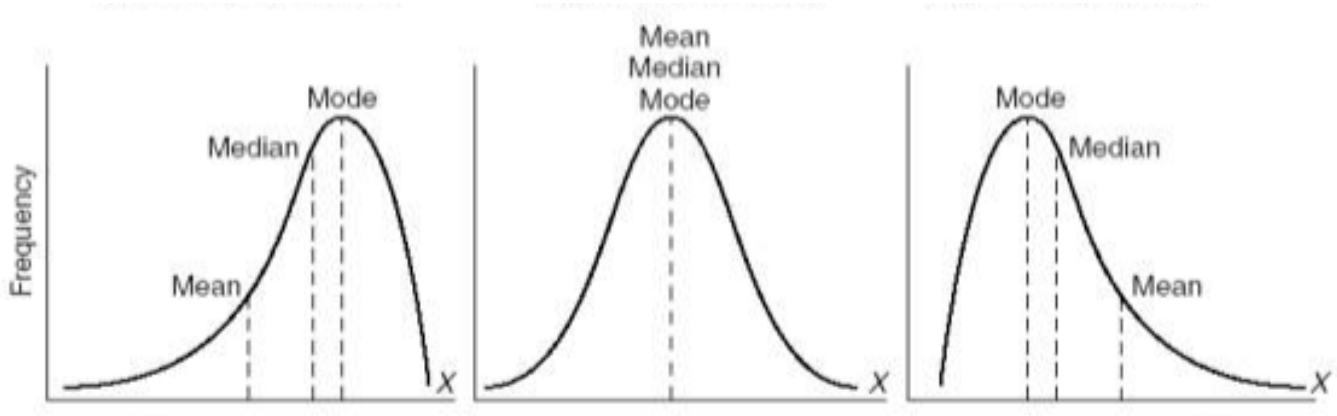
  [-5, -1, 0, **1**, **2**, 3, 8, 20]
  #elementos: 8
  La mediana es la media de los valores en las dos posiciones centrales
  Mediana = (1+2)/2 = 1.5

La **moda** es el valor que aparece con mayor frecuencia o más veces en la distribución.

Por ejemplo, la moda de [0,1,1,2,2,2,2,3,3,4,4,4,5] es 2.

La moda no es necesariamente única. Puede ocurrir que haya dos valores diferentes que sean los más frecuentes. Por ejemplo, para [10, 13, 13, 20, 20], tanto 13 como 20 son la moda.

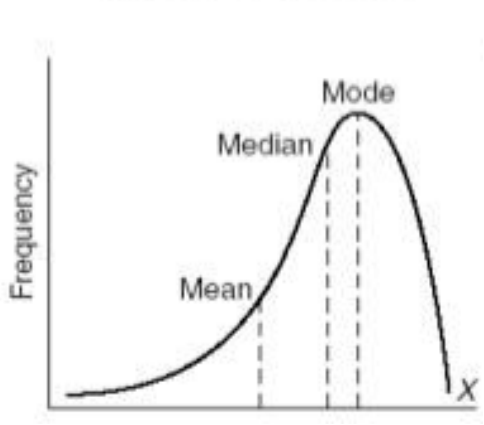
Nos referimos a la **asimetría** en cuanto a la distribución de los datos[1]:



- Una distribución con **asimetría a derecha** significa que la cola del lado derecho es más larga que la de la izquierda (gráfico a la derecha)
- De la misma manera, una distribución con **asimetría a izquierda**, significa que la cola de la izquierda es más larga que la de la derecha (gráfico a izquierda).
- Por último, una **distribución simétrica** no presenta este fenómeno dado que sus colas son de igual longitud al ser simétrica.

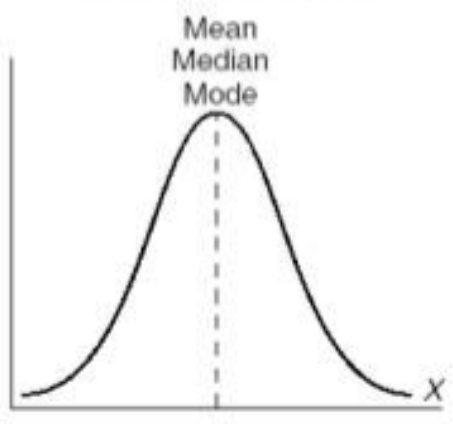
1: Estaremos hablando de asimetría en el contexto de distribuciones unimodales

La media, mediana y moda son afectadas por la asimetría:



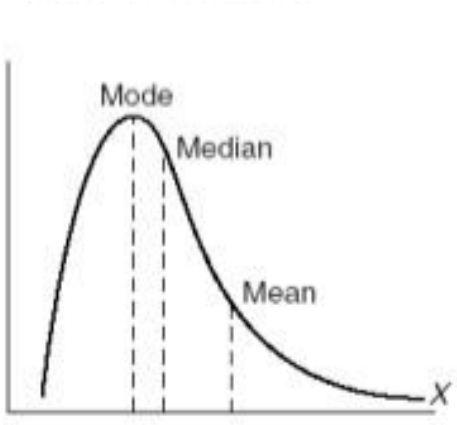
Asimetría a izquierda

Media < Mediana < Moda



Simetría

Media = Mediana = Moda



Asimetría a derecha

Moda < Mediana < Media

Las medidas de variabilidad indican cómo los datos están esparcidos. Nos vamos a focalizar en:

- **Rango**
- **Varianza**
- **Desvío estándar**
- **Coeficiente de variación**

Estas medidas proveen información complementaria (¡y no menos importante!) a las medidas de tendencia central (media, mediana y moda).

El **rango** es la diferencia entre el valor más bajo y más alto de la distribución.

En Python utilizamos la función numpy.ptp:

```
In [2]: import numpy as np
        a = np.array([2,3,6,7,8,11,4,6,17])
        np.ptp(a)

Out[2]: 15
```

El **rango intecuartil** es la diferencia entre el tercer y el primer cuartil de la distribución. Acumula el 50% de la distribución y, a diferencia del rango, es un estadístico robusto, es decir que se ve poco afectado por valores extremos.

La **varianza** es un valor numérico utilizado para describir cuánto varían los números de una distribución respecto a su media.

La varianza puede ser calculada como:

$$s^2 = \frac{\sum (X - \overline{X})^2}{N}$$

Esto es el **promedio de la diferencia elevada al cuadrado entre cada valor y la media**.

En Python utilizamos **numpy.var()** para calcular la varianza.

El **desvío estándar** es la raíz cuadrada de la varianza.

$$S = \sqrt{\frac{\Sigma(X - \overline{X})^2}{N}}$$

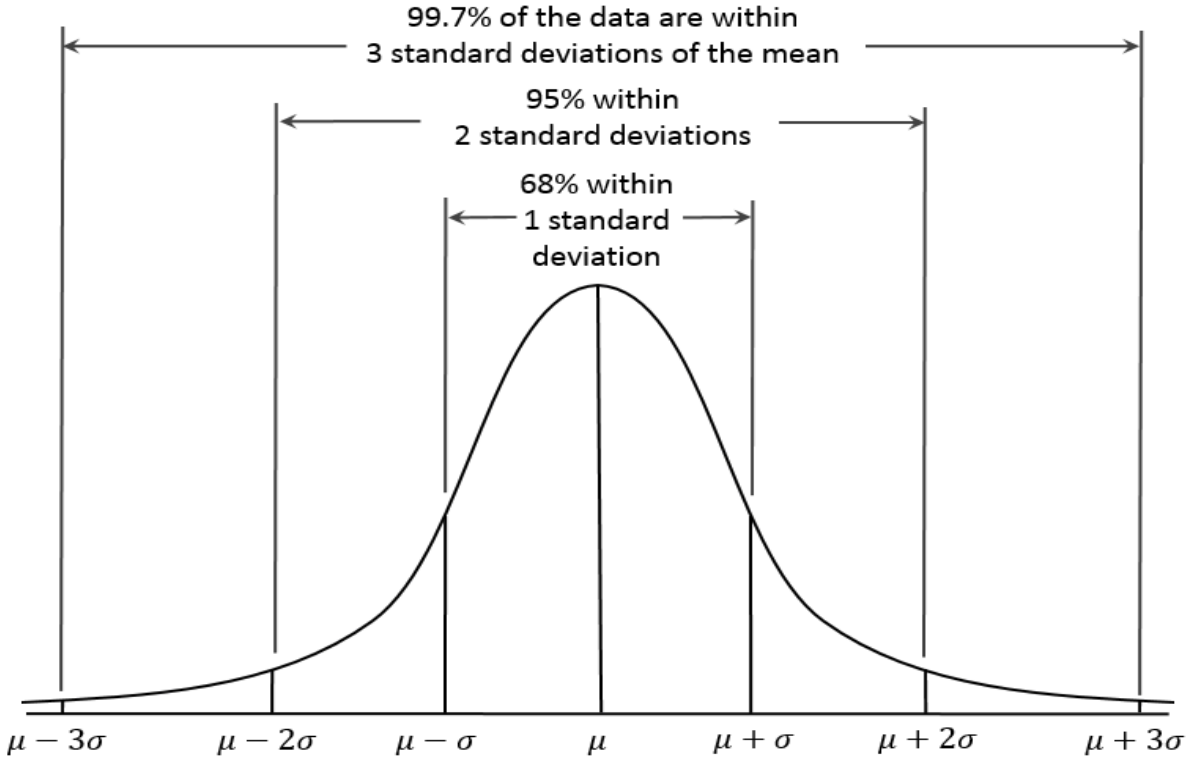
- El desvío es una medida de la dispersión de los datos

- NO ES la desviación promedio con respecto de la media. Como los desvíos están elevados al cuadrado los desvíos muy grandes cuentan más que proporcionalmente.

En Numpy:

```
std = np.std(n)
```

Una ventaja del desvío estándar es que está **expresada en las mismas unidades** que la distribución.
(En cambio, la varianza tiene otras unidades ya que está elevada al cuadrado.)



El **coeficiente de variación** es el desvío estándar dividido por la media

$$CV = \left(\frac{S}{\overline{X}} \right) \cdot 100\%$$

- El coeficiente de variación permite **comparar la dispersión de variables diferentes**.
  - Sirve si las variables tienen medias distintas.
  - También si las variables están expresadas en unidades distintas.

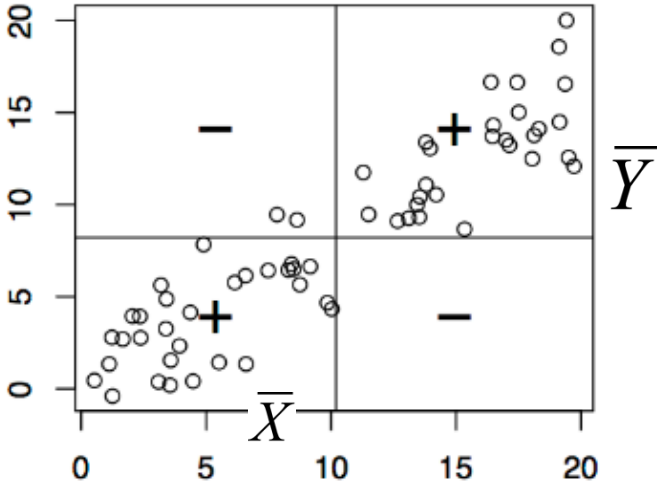
- El coeficiente de variación **no tiene unidades**.

— Covarianza

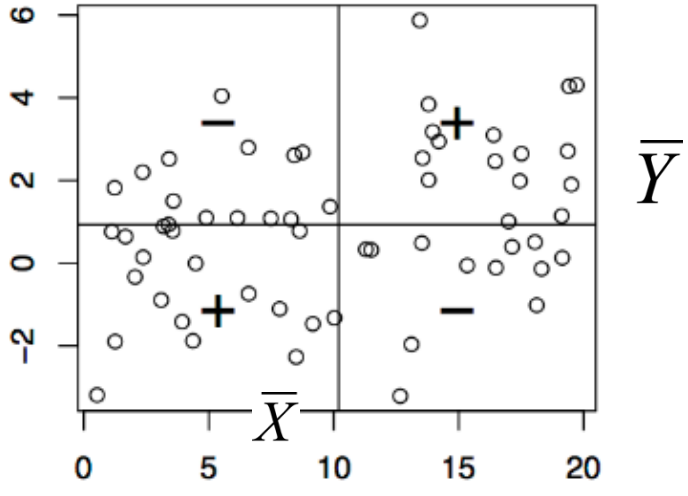
— Correlación

- Decimos que dos variables X e Y, tienen covarianza positiva cuando tienden a encontrarse por encima de su media al mismo tiempo y tienen covarianza negativa cuando al mismo tiempo, tienden a encontrarse una por debajo y otra por encima.
- En cambio X e Y tienen covarianza cercana a cero cuando las variables pueden encontrarse por encima o por debajo de su media *independientemente* de lo que haga la otra.
- La covarianza mide la relación lineal entre ambas variables, es decir, qué tanto se asemeja la relación con una función lineal.

Covarianza positiva



Covarianza cercana a cero



La covarianza se mide como:

$$cov_{XY} = \frac{1}{N} \sum_{i=1}^{N} (X - \overline{X})(Y - \overline{Y})$$

La covarianza de un conjunto de datos con p variables se puede representar con una matriz de p x p llamada **matriz de varianzas y covarianzas**:

1000*Covariances

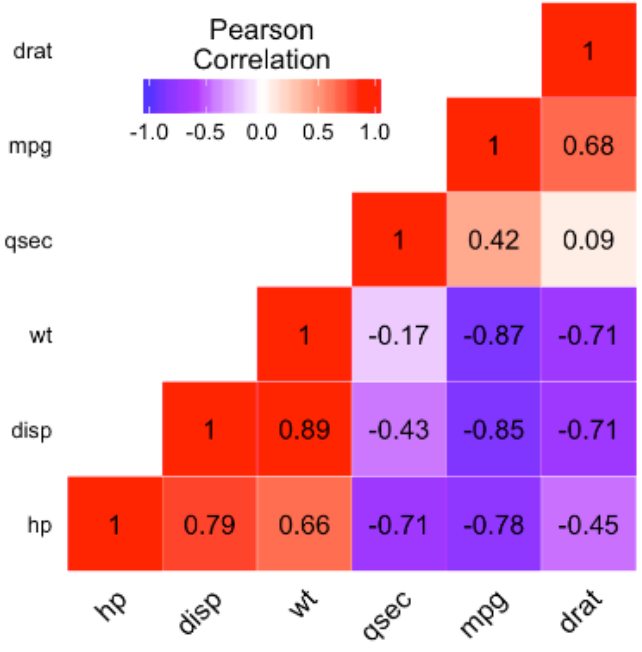
	^GSPC	^IXIC	XOM	C	GE	MSFT	K	GM
^GSPC	0.633	0.929	0.505	0.495	0.448	0.258	0.261	1.226
^IXIC	0.929	1.737	0.340	0.584	0.507	0.482	0.211	1.842
XOM	0.505	0.340	3.253	-0.421	-0.017	0.268	0.318	2.197
C	0.495	0.584	-0.421	1.923	0.688	0.176	0.277	-0.242
GE	0.448	0.507	-0.017	0.688	1.834	0.761	0.232	0.049
MSFT	0.258	0.482	0.268	0.176	0.761	1.945	0.181	1.315
K	0.261	0.211	0.318	0.277	0.232	0.181	1.045	0.688
GM	1.226	1.842	2.197	-0.242	0.049	1.315	0.688	9.429

* En la diagonal se encuentra la varianza de cada acción

* En el resto de la matriz se encuentran las covarianzas

— La correlación es una **versión estandarizada** (dividida por los desvíos estándar) de la covarianza:

$$r_{xy} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{X_i - \overline{X}}{S_x} \right) \left(\frac{Y_i - \overline{Y}}{S_y} \right)$$



* La correlación está acotada entre 1 y -1.

* Siempre que la covarianza es positiva, la correlación es positiva y viceversa.

* Mientras que la correlación no tiene unidades físicas, la covarianza sí.

PRÁCTICA GUIADA