# Ejercicios: NumPy básico

**Ejercicio 1.**

* Crea un array de 450 enteros aleatorios entre -10 y 10.
* Redimensiónalo como matriz de 5x70. ¿Hay algún problema? Si lo hay, corrígelo.
* Obtén con una función que encuentres en las bibliotecas de NumPy los datos no duplicados.
* Sobre los datos no duplicados, obtén la media y los cuartiles, buscando las funciones adecuadas.
* Dibuja el histograma y los cuartiles del array. Nota: para dibujar líneas se pueden utilizar funciones como vlines().

**Ejercicio 2.**

* Crea dos arrays de 100 elementos con números aleatorios de distribuciones diferentes, pero en un rango de valores similar.
* Dibuja sus histogramas y compáralos.
* ¿Con qué método estadístico podrías contrastar si las dos muestras provienen de la misma distribución? ¿En qué paquete del stack científico de Python podría estar?

**Ejercicio 3.**

* Si no lo conocías, lee el ejemplo de qué es una media móvil aquí: <http://www.investopedia.com/terms/m/movingaverage.asp> (consideramos solo las medias móviles simples)
* Genera un array de supuesta cotizaciones de un valor de la bolsa duante un período. Nota: Puedes simularlo con un “random walk” a partir de un valor de cotización inicial
* Busca una función en el stack científico de Python que implemente la media móvil y aplícalo a los datos generados.
* (\*\*) Ahora haz tu propia implementación de la media móvil utilizando arrays de NumPy.
* Compara la eficiencia de tu versión y la de NumPy.
* (\*\*) Investiga cómo obtener datos reales de un valor de la bolsa a partir de Yahoo Financials (si no lo consigues por programa, puedes descargarlo a un fichero a mano y leerlo después), y aplica la media móvil.

**Ejercicio 4. Entrada/salida de arrays NumPy (\*)**

* NumPy tiene una categoría de funciones para grabar y recuperar arrays de disco: <https://docs.scipy.org/doc/numpy/reference/routines.io.html>

A partir del siguiente código:

import numpy as np

N = 10

a = np.random.uniform(size=N)

b = np.random.normal(size=N)

b

array([ 0.52284741, 0.23829581, -0.49767523, 0.54278876, 0.27790099,

1.94402675, -0.3543395 , 0.57639513, -0.80058146, -0.89837096])

* Guardar los arrays anteriores en formato específico de NumPy comprimido y formato de texto (sin comprimir).
* Comparar las diferencias de espacio en disco desde el mismo Notebook, y ver sus contenidos con el comando head del sistema operativo.
* Cargarlos de nuevo en memoria con nombres de variable diferentes.
* (opcional) Guardar los dos arrays en un solo fichero de texto y recuperarlos de nuevo. Pista: utilizar tuplas.

1. **Generación de datos - distribuciones de SciPy (\*\*)**

SciPy tiene un buen número de distribuciones de probabilidad conocidas implementadas, continuas y discretas: <https://docs.scipy.org/doc/scipy/reference/stats.html>

Por ejemplo, podemos trabajar con distribuciones normales de la siguiente forma.

from scipy.stats import norm

# Creamos una distribución normal con media 5.

mi\_normal = norm(5)

# Obtenemos los percentiles de los extremos de la distribución:

x = np.linspace(mi\_normal.ppf(0.001), mi\_normal.ppf(0.999), 100)

# Obtenemos los estadísticos comunes:

m, v = mi\_normal.stats(moments='mv')

print(m, v)

# Dibujamos la función

%matplotlib inline

import matplotlib.pyplot as plt

plt.plot(x, mi\_normal.pdf(x), 'r-', lw=3)

(array(5.0), array(1.0))



Podemos generar a su vez datos aleatorios para simular una muestra de la siguiente manera.

r = mi\_normal.rvs(size=1000)

plt.hist(r, normed=True, alpha=0.2, bins="auto")

(array([ 0.00381833, 0.00381833, 0.00381833, 0.011455 , 0.03054667,

0.04200167, 0.08782167, 0.10691334, 0.194735 , 0.19091667,

0.26346501, 0.33601334, 0.32074001, 0.42383501, 0.44292668,

0.37801501, 0.27110167, 0.24437334, 0.17946167, 0.11836834,

0.057275 , 0.06491167, 0.03054667, 0.011455 ]),

array([ 1.44157533, 1.70346969, 1.96536405, 2.22725841, 2.48915278,

2.75104714, 3.0129415 , 3.27483586, 3.53673023, 3.79862459,

4.06051895, 4.32241331, 4.58430768, 4.84620204, 5.1080964 ,

5.36999077, 5.63188513, 5.89377949, 6.15567385, 6.41756822,

6.67946258, 6.94135694, 7.2032513 , 7.46514567,

7.72704003]),

<a list of 24 Patch objects>)



Queremos ahora estudiar la distribución de Pareto. Podemos ver su forma en Wikipedia:

https://es.wikipedia.org/wiki/Distribución\_de\_Pareto

Se pide lo siguiente:

* Representar las funciones de densidad y de probabilidad similares a las de Wikipedia.
* Hallar el percentil del 80% con k=1. Razonar sobre el Principio de Pareto, se puede consultar aquí: <https://en.wikipedia.org/wiki/Pareto_principle>

1. **Generación de datos - otras distribuciones**

Si no encontrásemos la distribución en las bibliotecas de ScyPy pero la conocemos ("sabemos la fórmula"), siempre podemos generar los datos aleatorios generando un array con los datos en el rango en que está definida, y después tomando aleatoriamente un número de elementos de allí ("con

reemplazo").

Vamos a pensar si tenemos esta función: f(t) definida por:

http://quicklatex.com/cache3/1f/ql_53e953981ae505c01171a12f912fc51f_l3.png

Se pide:

* Dibujar la función en el rango [0, 10].
* Generar una muestra aleatoria de la misma.
* Dibujar esa muestra y su histograma.