S02 T04

March 30, 2022

1 S02 T04: Pràctica amb programació numèrica

1.1 Luis Pardina - Data Science - 30/03/2022

Exercici 1: Crea una funció que donat un Array d'una dimensió, et faci un resum estadístic bàsic de les dades. Si detecta que l'array té més d'una dimensió, ha de mostrar un missatge d'error

[1]:
```
import numpy as np
from numpy import random

#genero una array d'1 dimensió amb l'ajuda de random amb n sencers entre 0 i 49
n = random.randint(10,20)   # n entre 10 i 19
matriu = np.array(random.randint(50, size=(n)))

matriu_dolenta = np.array([[1,2],[3,4]])

def resum(arr):
    if arr.ndim > 1:
        print("\nLa matriu te més d'1 dimensió!!\n")
    else:
        num_elem = len(arr)
        maxim = arr.max()
        minim = arr.min()
        mitja = np.mean(arr)
        mitjana = np.median(arr)
        desviacio = arr.std()
        print("Matriu 1d amb {0} elements, max = {1}, min = {2}".
 ↪format(num_elem, maxim, minim))
        print("Mitja = {0:1f}, Mitjana = {1:1f}, Desvest = {2:1f}".
 ↪format(mitja, mitjana, desviacio))

resum(matriu)
resum(matriu_dolenta)
```

```
Matriu 1d amb 11 elements, max = 49, min = 5
Mitja = 22.000000, Mitjana = 18.000000, Desvest = 12.898203

La matriu te més d'1 dimensió!!
```

Exercici 2: Crea una funció que et generi un quadrat NxN de nombres aleatoris entre el 0 i el 100

[2]:
```
dimensio = random.randint(5,11)   # n entre 5 i 10
def quadrat(n):
    return np.random.randint(0, 100, (n,n))

print(quadrat(dimensio))
```

```
[[23  3 85  3 49 17 85 94 43]
 [87 49 19 92  9 32 64 44 61]
 [95 94 70 97 97 67 39 93 98]
 [21 88 47 96 54 20 99  2 84]
 [15 20 47 71 56 70 22 50 22]
 [34 20 39 17 90 95 84 49 69]
 [24 68 17 43 36 34 67 36 43]
 [68 56 24 60 25  2 84 84 97]
 [59  0 45 41 67 59 90 75 79]]
```

Exercici 3: Crea una funció que donada una taula de dues dimensions, et calculi els totals per fila i els totals per columna.

[3]:
```
test = np.random.randint(0, 100, (3,3))
print(test)

def totals_fila_i_columna(arr):
    totfil = arr.sum(axis=0)              #mètode sum axis 0
    totcol = arr.sum(axis=1)              #mètode sum axis 1
    return totfil, totcol

a,b = totals_fila_i_columna(test)
print("Sumes columnes: {0}, sumes files: {1}".format(a,b))
```

```
[[96 58 51]
 [30 39  2]
 [80 97 66]]
Sumes columnes: [206 194 119], sumes files: [205  71 243]
```

Exercici 4: Implementa manualment una funció que calculi el coeficient de correlació. Informa't-en sobre els seus usos i interpretació.

[4]:
```
a = np.array([2,4,6,8,10,12,14,16,18,20])
b = np.array([4.85,9.15,12.75,17.25,20.95,25.05,28.75,33.25,36.85,41.15])

def correl(data1,data2):
    avx = data1.mean() #mitja de primera array
    avy = data2.mean() #mitja segona array

    x = data1 - avx    #x contindrá ara el valors menys la mitja
    y = data2 - avy    #y contindrà ara els valors menys la mitja
```

```
    sxy = sum(x*y)/(len(x)-1)           #covariança x i y
    sx = (sum(x*x)/(len(x)-1)) ** 0.5   #variança x
    sy = (sum(y*y)/(len(y)-1)) ** 0.5   #variança y

    return sxy / (sx * sy)              #correlació

print(correl(a,b))
print(np.corrcoef(a,b))
```

```
0.9998733131178795
[[1.         0.99987331]
 [0.99987331 1.        ]]
```

[ ]: