M3_T01

January 13, 2023

1 Sprint 3

1.1 Tasca M3 T01

1.1.1 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
import statistics as st

def resest(array):
    suma=[]
    res=[]
    dif=[]
    avg=[]
    mult=[]
    if np.ndim(array) == 1:
        suma=np.sum(array)
        avg=st.mean(array) #o bé avg=suma/len(arr)
        dif=np.diff(array)

        ngran=max(array)
        npetit=min(array)
        res=ngran-npetit

        mult=1
        for jj in range(len(array)):
            mult=mult*array[jj]
        print('\033[1m'+"Resum estadístic"+'\033[0m')
        print("\nSuma:",suma)
        print("Resta més gran amb més petit:",res)
        print("Diferència entre valors:",dif)
        print("Mitjana:",avg)
        print("Mult. entre valors (5!):",mult)
    else:
        print('Error, not a 1-dim array')
```

```
    return suma,res,dif,avg,mult


#Operacions amb la array

arr=np.array([3,4,3,6,9,5])
suma,res,dif,avg,mult=resest(arr)
```

```
Resum estadístic

Suma: 30
Resta més gran amb més petit: 6
Diferència entre valors: [ 1 -1  3  3 -4]
Mitjana: 5
Mult. entre valors (5!): 9720
```

1.1.2 Exercici 2

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

[2]:
```
from numpy import random

def sqrand(N):
    randmat=random.randint(100, size=(N,N))
    print(randmat)


#Example

N=10
sqrand(N)
```

```
[[ 2 59 27  8 33 54 51 42  0 22]
 [79 31 67  0 11 82 70 23 41 28]
 [30 88 13 63 94 36 79 96 62 69]
 [99 44 94 11 42 89 10  2 61 14]
 [54 71 48  7 25 98 83 26 45 52]
 [49  6 21 59 64 90 32 25 57  5]
 [55  3 49  8 93 12 35 22 90 84]
 [54 71 11 54 67 64 66 46 97 36]
 [66 79 36 10 40 33 20 99 29 61]
 [68 62 96 43  9 53 74 79 90 17]]
```

1.1.3 Exercici 3

Crea una funció que donada una taula de dues dimensions (NxM), et calculi els totals per fila i els totals per columna.

[3]:
```
def crtots(N,M):
    randmat=random.randint(50, size=(N,M))
    print('Matrix ->')#matriu NxM
    print(randmat) #fila en blanc per separar

    files=np.sum(randmat, axis=1)
    cols=np.sum(randmat, axis=0)
    print('\nTOTAL PER FILES:',*files)
    print('TOTAL PER COLUMNES:',*cols)


#Example

N=4
M=6
crtots(N,M)
```

```
Matrix ->
[[24 45 28 36 38 26]
 [32 18 44 33  6 24]
 [ 5 13 18 35 22 12]
 [37 32 35  7 29 30]]

TOTAL PER FILES: 197 157 105 170
TOTAL PER COLUMNES: 98 108 125 111 95 92
```

1.1.4 Exercici 4

Implementa manualment una funció que calculi el coeficient de correlació. Informa-te'n sobre els seus usos i interpretació.

[5]:
```
def calcc(arr1,arr2):
    altm=st.mean(arr1)
    pesm=st.mean(arr2)
    num=0
    den=0

    for jj in range(len(arr1)):
        num=num+(arr1[jj]-altm)*(arr2[jj]-pesm)
        den=den+np.sqrt(np.square(arr1[jj]-altm)*np.square(arr2[jj]-pesm))

    r=num/den
    rnp=np.corrcoef(arr1,arr2) # el coeficient de correlació és el que està a␣
↪les diagonals de la matriu
    print('Coeficient de correlació a mà:',r)
    print('Coeficient de correlació amb numpy:',rnp[0,0])
    return r,rnp
```

```
#Example

arr1=np.array([1.75, 1.88, 1.65, 1.72, 1.93, 1.83])
arr2=np.array([69, 73, 56, 63, 105, 80])

r,rnp=calcc(arr1,arr2)
```

```
Coeficient de correlació a mà: 0.9787234042553193
Coeficient de correlació amb numpy: 0.9999999999999999
```

Podem veure com el coeficient és pràcticament r=1, el que indica que aquestes dos variables estàn lligades entre sí linealment. Quant més gran sigui una, més creixerà l'altre per norma general.