S03-T04-Programacio-Numerica

October 18, 2021

0.1 S03 T04: Pràctica amb programació numèrica

Descripció
Familiaritza't amb la Programació Numèrica a través de la llibreria NumPy.

- **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.
- **Exercici 2**
  Crea una funció que et generi un quadrat NxN de nombres aleatoris entre el 0 i el 100.
- **Exercici 3**
  Crea una funció que donada una taula de dues dimensions, et calculi els totals per fila i els totals per columna.
- **Exercici 4**
  Implementa manualment una funció que calculi el coeficient de correlació. Informa't-en sobre els seus usos i interpretació.

- **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.

[50]:
```python
import numpy as np

#Function to make a statistical summary
#Takes as argument an array and returns a dictionarywith the data
def statisticsSummary(array):
    if array.ndim == 1:

        maxNum = np.max(array)
        minNum = np.min(array)
        arrayMean = np.mean(array)
        arrayMedian = np.median(array)
        arrayStandDev = np.std(array)
        arrayVariance = np.var(array)

        statisticSummary = {"maxNum": maxNum,
                           "minNum" : minNum,
                           "arrayMean" : arrayMean,
                           "arrayMedian" :arrayMedian,
                           "arrayStandDev" : arrayStandDev,
                           "arrayVariance" : arrayVariance}

        return statisticSummary

    else:
        print("Error!! the dimension of the array is different than 1!!")
```

[ ]:
```
myArray = np.array([1,3,5,3,56,3,5,6,7,8,45,24,36]) #Creating an array

#We call and print the statistical summary
print("A statistical summary of the array:", statisticsSummary(myArray))
```

I check that the control of 1 dimension works:

[16]:
```
my2ndArray = np.random.randint(10, size=(3, 4))  # Two-dimensional array
print(my2ndArray)

#We call and print the statistical summary
print("A statistical summary of the array:", statisticsSummary(my2ndArray))
```

```
[[3 6 7 3]
 [7 3 9 8]
 [3 4 4 8]]
Error!! the dimension of the array is different than 1!!
A statistical summary of the array: None
```

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

[17]:
```
def squareMatrix(n):
    squareMatrix = np.random.randint(0,101, size = (n,n))
    return squareMatrix
```

[19]:
```
mySquareMatrix = squareMatrix(7) #Calling the function to generate a matrix 7 x␣
 ↪7

print(mySquareMatrix)
print(mySquareMatrix.shape) #Check the shape to see if the function works␣
 ↪correctly
```

```
[[94 38 13 50 58 66 33]
 [12 53  9 35 55 61 25]
 [67 56 12 33 81 24 49]
 [44 88 28 88 68 46 10]
 [47 49 52  8 79 73 85]
 [72 10 97 32 56 42  2]
 [61 44 94 31  7 98 79]]
(7, 7)
```

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

[31]:
```
#Function calculating the row and column totals, for any matrix given
def sumRowColumns(matrix):
    #I creat 2 lists to save the totals per row, and columns
    matrixColumns = []
    matrixRows = []
    for i in range((matrix.ndim)): #loop to sum each row of the matrix
        sumRow = np.sum(matrix[i,:])
        matrixRows.append(sumRow)
    for j in range((matrix.ndim)): #loop to sum each column of the matrix
        sumColumns = np.sum(matrix[:,j])
        matrixColumns.append(sumColumns)
    print("Row totals: ", matrixRows)
    print("Column Totals: ", matrixColumns)
    return matrixRows, matrixColumns #I think this function needs to return the␣
 ↪values
```

[32]:
```
myMatrix2x2 = np.random.randint(0, 11,(2,2))
print(myMatrix2x2)
sumRowColumns(myMatrix2x2)
```

```
[[0 6]
 [6 2]]
Row totals:  [6, 8]
Column Totals:  [6, 8]
```

[32]:
```
([6, 8], [6, 8])
```

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

numpy.corrcoef

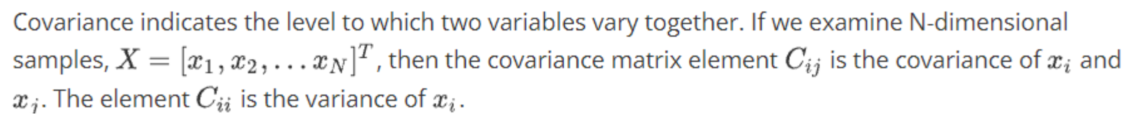
Return Pearson product-moment correlation coefficients. The relationship between the correlation coefficient matrix, R, and the covariance matrix, C, is

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} * C_{jj}}}$$

The values of R are between -1 and 1, inclusive.

numpy.cov

Estimate a covariance matrix, given data and weights.


Covariance indicates the level to which two variables vary together. If we examine N-dimensional samples, $X = [x_1, x_2, \ldots x_N]^T$, then the covariance matrix element C_{ij} is the covariance of x_i and x_j. The element C_{ii} is the variance of x_i.

See the notes for an outline of the algorithm.

[44]:
```
import math
def correlationCoef(matrix):
    CovMatrix = np.cov(matrix)
    cc = np.empty(matrix.shape) #creats an empty ndarray with the sameshape as␣
 ↪the entry matrix
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[0]):
            #Calculates each element of the cc matrix
            cc[i,j] = (CovMatrix[i,j])/math.
 ↪sqrt((CovMatrix[i,i])*CovMatrix[j,j])
    return cc
```

[39]:
```
mmms = np.random.randint(1,11,(10,10))
print(mmms)
print(mmms.ndim)
print(mmms.shape)
```

```
[[ 8  2  2  4  8  7  2  2  9 10]
 [ 7  6  8  6  3  4  9  2  4  7]
 [ 8  1  9  2  3  9  1  2  3  6]
 [ 7 10  5  8  6  7  6  3 10  8]
 [ 5 10 10  6  9  3  7 10  9  5]
 [ 7  6  8  1  5  6  6  8  8 10]
 [ 1  3  2 10  6  5  7 10 10  3]
 [ 9  2  2  9 10  7 10  1 10  9]
 [ 4  3  5  7  5  1  8  7  9  5]
 [ 3  6  9  4  5  1  8  9  9  6]]
2
(10, 10)
```

[48]:
```
print(correlationCoef(mmms))
print(correlationCoef(mmms).ndim)
print(correlationCoef(mmms).shape)
```

```
[[ 1.         -0.21311104  0.35320137  0.38888198 -0.49600251  0.26434823
  -0.12410794  0.67111071 -0.10584014 -0.38557015]
 [-0.21311104  1.          0.17344448  0.15856758 -0.19978732  0.04052204
  -0.51113631  0.15237642  0.03244855  0.07018624]
 [ 0.35320137  0.17344448  1.         -0.15715812 -0.47446596  0.35141625
 -0.6249846  -0.04192162 -0.57124429 -0.38954931]
 [ 0.38888198  0.15856758 -0.15715812  1.         -0.16153213 -0.12777531
  -0.07420489  0.35897771 -0.06394859 -0.22131333]
 [-0.49600251 -0.19978732 -0.47446596 -0.16153213  1.          0.12641889
   0.20388991 -0.53392124  0.40492627  0.78201406]
 [ 0.26434823  0.04052204  0.35141625 -0.12777531  0.12641889  1.
  -0.36503978 -0.22846036 -0.01906579  0.42888877]
 [-0.12410794 -0.51113631 -0.6249846  -0.07420489  0.20388991 -0.36503978
   1.          0.15232781  0.6802859   0.29889007]
 [ 0.67111071  0.15237642 -0.04192162  0.35897771 -0.53392124 -0.22846036
   0.15232781  1.          0.26452511 -0.33153658]
 [-0.10584014  0.03244855 -0.57124429 -0.06394859  0.40492627 -0.01906579
   0.6802859   0.26452511  1.          0.70999255]
 [-0.38557015  0.07018624 -0.38954931 -0.22131333  0.78201406  0.42888877
   0.29889007 -0.33153658  0.70999255  1.        ]]
2
(10, 10)
```

[49]:
```
print(np.corrcoef(mmms))
print(np.corrcoef(mmms).ndim)
print(np.corrcoef(mmms).shape)
```

```
[[ 1.         -0.21311104  0.35320137  0.38888198 -0.49600251  0.26434823
  -0.12410794  0.67111071 -0.10584014 -0.38557015]
 [-0.21311104  1.          0.17344448  0.15856758 -0.19978732  0.04052204
  -0.51113631  0.15237642  0.03244855  0.07018624]
 [ 0.35320137  0.17344448  1.         -0.15715812 -0.47446596  0.35141625
  -0.6249846  -0.04192162 -0.57124429 -0.38954931]
 [ 0.38888198  0.15856758 -0.15715812  1.         -0.16153213 -0.12777531
  -0.07420489  0.35897771 -0.06394859 -0.22131333]
 [-0.49600251 -0.19978732 -0.47446596 -0.16153213  1.          0.12641889
   0.20388991 -0.53392124  0.40492627  0.78201406]
 [ 0.26434823  0.04052204  0.35141625 -0.12777531  0.12641889  1.
  -0.36503978 -0.22846036 -0.01906579  0.42888877]
 [-0.12410794 -0.51113631 -0.6249846  -0.07420489  0.20388991 -0.36503978
   1.          0.15232781  0.6802859   0.29889007]
 [ 0.67111071  0.15237642 -0.04192162  0.35897771 -0.53392124 -0.22846036
   0.15232781  1.          0.26452511 -0.33153658]
 [-0.10584014  0.03244855 -0.57124429 -0.06394859  0.40492627 -0.01906579
   0.6802859   0.26452511  1.          0.70999255]
 [-0.38557015  0.07018624 -0.38954931 -0.22131333  0.78201406  0.42888877
   0.29889007 -0.33153658  0.70999255  1.        ]]
2
(10, 10)
```

We can see they are the same matrix!! iujuu

0.1.1 Correlation

https://realpython.com/numpy-scipy-pandas-correlation-python/

Correlation coefficients quantify the association between variables or features of a dataset. These statistics are of high importance for science and technology, and Python has great tools that you can use to calculate them. SciPy, NumPy, and Pandas correlation methods are fast, comprehensive, and well-documented.

Statistics and data science are often concerned about the relationships between two or more variables (or features) of a dataset.

Note: When you're analyzing correlation, you should always have in mind that **correlation does not indicate causation**. It quantifies the strength of the relationship between the features of a dataset. Sometimes, the association is caused by a factor common to several features of interest.

There are several statistics that you can use to quantify correlation.
- Pearson's r
- Spearman's rho
- Kendall's tau

Pearson's coefficient measures linear correlation, while the Spearman and Kendall coefficients compare the ranks of data. There are several NumPy, SciPy, and Pandas correlation functions and methods that you can use to calculate these coefficients. You can also use Matplotlib to conveniently illustrate the results.

corrcoef() returns the correlation matrix
The values on the main diagonal of the correlation matrix (upper left and lower right) are equal to 1. The upper left value corresponds to the correlation coefficient for x and x, while the lower right value is the correlation coefficient for y and y. They are always equal to 1.

However, what you usually need are the non-diagonal values of the correlation matrix. These values are equal and both represent the Pearson correlation coefficient for x and y.

Linear Correlation Linear correlation measures the proximity of the mathematical relationship between variables or dataset features to a linear function. If the relationship between the two features is closer to some linear function, then their linear correlation is stronger and the absolute value of the correlation coefficient is higher.

Pearson Correlation Coefficient Consider a dataset with two features: x and y. Each feature has n values, so x and y are n-tuples. Say that the first value x from x corresponds to the first value y from y, the second value x from x to the second value y from y, and so on. Then, there are n pairs of corresponding values: (x , y), (x , y), and so on. Each of these x-y pairs represents a single observation.

The **Pearson (product-moment) correlation coefficient** is a measure of the linear relationship between two features. It's the ratio of the covariance of x and y to the product of their standard deviations. It's often denoted with the letter r and called Pearson's r. You can express this value mathematically with this equation:

$r = \Sigma\ ((x\ - \text{mean}(x))(y\ - \text{mean}(y)))\ (\sqrt{}\Sigma\ (x\ - \text{mean}(x))^2\ \sqrt{}\Sigma\ (y\ - \text{mean}(y))^2)\ ^1$

Here are some important facts about the Pearson correlation coefficient:

- The Pearson correlation coefficient can take on any real value in the range -1 r 1.
- The maximum value $r = 1$ corresponds to the case in which there's a perfect positive linear relationship between x and y. In other words, larger x values correspond to larger y values and vice versa.
- The value $r > 0$ indicates positive correlation between x and y.
- The value $r = 0$ corresponds to the case in which there's no linear relationship between x and y.
- The value $r < 0$ indicates negative correlation between x and y.
- The minimal value $r = -1$ corresponds to the case when there's a perfect negative linear relationship between x and y. In other words, larger x values correspond to smaller y values and vice versa.

Pearson's r Value	Correlation Between x and y
equal to 1	perfect positive linear relationship
greater than 0	positive correlation
equal to 0	no linear relationship
less than 0	negative correlation
equal to -1	perfect negative linear relationship

In short, a larger absolute value of r indicates stronger correlation, closer to a linear function. A smaller absolute value of r indicates weaker correlation.

```
[ ]:
```