

Louis  
ANDROS-BESSEDE  
HPSI 1

## DS 9 Informatique

- 1) 

```
SELECT idpatient FROM medical
WHERE etat = "hernie discale";
```
- 2) 

```
SELECT nom, prenom FROM patient
JOIN medical
ON patient.id = medical.idpatient
WHERE etat = "spondylolisthésis";
```
- 3) 

```
SELECT etat, COUNT(idpatient) FROM medical
GROUP BY etat;
```
- 4) La bibliothèque de mumpy contient plus d'outils pour traiter les tableaux que Python, on peut donc traiter les vecteurs plus facilement.



5) On a  $N \times 6$  "informations" à stocker  
sur 4 octets car codés sur 32 bits.

donc  $4 \times 6 \times 100\,000 = 2,4$  Mo de stockage

```
6) def separation_pargroupes(data, état):  
    groupe = [ [], [], [] ]  
    for i in range(len(data)):  
        groupes[état[i]] . append(data[i])  
  
    return groupe
```

7)

8)



9)

$$\text{On pose } a_{\text{norm}ij} = \frac{a_{ij}}{\sqrt{\min(X)^2 + \max(X)^2}}$$

10) `def min_max(X):`

`min = X[0]`

`max = X[0]`

`for i in range(len(X):`

`if X[i] < min:`

`min = X[i]`

`if X[i] > max:`

`max = X[i]`

`return min, max.`

11) On veut d'abord normaliser les données.

`def extreminums(data):`

`ext = [[] for i in range(len(data))]:`

`for i in range(len(data):`

`ext[i].append(min_max(data[i]))`

`return ext.`

Cette fonction est censée récupérer les extrêmes de chaque famille de vecteurs du tableau



~~def normalise (data):
 """ en utilisant la proposition de """
 data\_n = [ [] for i in range len(data)]
 for i in range len(data):
 for j in range len(data):
  $nm = data[i][j] * e$~~

def normalise (data):
 ~~data\_n = [ [] for i in range len(data)]~~
 ext = extremes (data)
 max = [ data[i][1] for i in range len(ext) ]
 min = [ data[i][0] for i in range len(ext) ]
 for i in range len(data):
 for j in range len(data):
  $nm = data[i][j] * \sqrt{min[i]**2 + max[i]**2}$ 
~~data\_n.append~~
~~data\_n.append~~
 data\_n[i].append (nm)
 return data\_n.

def distance (z, data):
 data\_n = normalise (data)
 ~~distance = [ ]~~
 for i in range len(z):
 ~~for j in range len(data):~~
 $dist = \sqrt{z[i]**2 - data_n[i]**2}$ 
 distance.append (dist)
 return distance.



15) def moyenne(x):  
 total = len(x)  
 somme = sum(x)  
 return somme / total

def variance(x):  
 total = len(x)  
 somme = 0  
 moy = moyenne(x)  
 for i in range(len(x)):  
 somme += x[i] - moy\*\*2 + somme  
 return somme / n

16)

✓