TP : Comparaison entre `np.sum` et `np.einsum` sur un dataset NBA

Ammar Djebabla

November 28, 2025

Objectifs

Dans ce TP, vous allez :

- Charger un dataset de statistiques NBA.
- Effectuer des sommes sur plusieurs colonnes avec `np.sum`.
- Utiliser `np.einsum` pour effectuer les mêmes calculs.
- Comparer les temps d'exécution entre les deux méthodes.

1 Préparation du dataset

Pour cet exercice, nous avons un dataset simplifié avec les colonnes suivantes :

Nom de la colonne	Type	Exemple
player	string	"LeBron James"
team	string	"LAL"
POS	string	"SF"
GP	int64	82
MIN	float64	34.7
PTS	float64	27.1
REB	float64	7.0
AST	float64	11.7
STL	float64	2.8
BLK	float64	3.7
TO	float64	5.7
salary	int64	20000000

2 Chargement du dataset en Python

```
import numpy as np
import pandas as pd
import time

# Exemple de dataset simul
data = {
    "PTS": np.random.rand(1000000) * 30,  # 1 million de points
```

```
    "REB": np.random.rand(1000000) * 15,
    "AST": np.random.rand(1000000) * 12,
}

df = pd.DataFrame(data)

# Conversion en array NumPy pour acc l rer les calculs
arr = df.to_numpy()
```

3 Somme avec `np.sum`

```
start = time.time()

# Somme des points, rebonds et passes
total_sum = np.sum(arr, axis=0)

end = time.time()
print("R sultat np.sum:", total_sum)
print("Temps d'ex cution np.sum:", end - start, "secondes")
```

4 Somme avec `np.einsum`

```
start = time.time()

# np.einsum permet de sommer les colonnes facilement
total_einsum = np.einsum('ij->j', arr)

end = time.time()
print("R sultat np.einsum:", total_einsum)
print("Temps d'ex cution np.einsum:", end - start, "secondes")
```

5 Analyse des résultats

- Comparez les valeurs renvoyées par `np.sum` et `np.einsum`. Elles doivent être identiques.
- Comparez les temps d'exécution. Pour de grands datasets, `np.einsum` peut être légèrement plus rapide ou plus flexible pour des calculs complexes (ex : somme pondérée, produit matriciel).

6 Exercice avancé : somme pondérée

On souhaite maintenant calculer la somme pondérée des colonnes PTS, REB et AST.

```
weights = np.array([0.5, 1.5, 2.0])  # pond rations pour PTS, REB, AST

# M thode 1 : avec np.sum
start = time.time()
weighted_sum1 = np.sum(arr * weights, axis=1)
end = time.time()
print("R sultat np.sum pond r :", weighted_sum1[:5])  # affichage des 5
    premiers
```

```
print("Temps d'ex cution np.sum pond r :", end - start, "secondes")

# M thode 2 : avec np.einsum
start = time.time()
weighted_sum2 = np.einsum('ij,j->i', arr, weights)
end = time.time()
print("R sultat np.einsum pond r :", weighted_sum2[:5])
print("Temps d'ex cution np.einsum pond r :", end - start, "secondes")

# V rification que les deux r sultats sont identiques
print("Les deux r sultats sont identiques :", np.allclose(weighted_sum1,
    weighted_sum2))
```

- Comparez les temps d'exécution des deux méthodes.
- Observez que `np.einsum` peut être plus rapide pour des opérations vectorisées sur de grands datasets.
- Notez que `np.allclose` permet de vérifier que les deux méthodes donnent les mêmes résultats, même si de légères différences de précision apparaissent.