NumPy pour l’analyse statistique

Maîtriser NumPy : Le compagnon indispensable pour les Data Lovers · https://moncoachdata.com/tutos/numpy-pour-lanalyse-statistique/

L'**analyse statistique avec NumPy** repose sur quelques opérations simples : transformer les données (centrage, standardisation, logarithme), prélever des échantillons aléatoires et générer des nombres aléatoires selon une distribution donnée.

Transformer les données

NumPy n'a pas de fonction dédiée à chaque transformation, mais ses fonctions de base suffisent à les réaliser :

- **Centrage :** on soustrait la moyenne à chaque point de données. On l'utilise souvent pour supprimer l'effet d'un terme constant ou faciliter la convergence d'un modèle ;
- **Standardisation :** on transforme les données pour qu'elles aient une moyenne de 0 et un écart type de 1, ce qui facilite la comparaison de données d'échelles différentes ;
- **Transformation logarithmique :** elle rend les données plus symétriques ou stabilise la variance en cas de croissance exponentielle.

```
import numpy as np

data = np.array([10, 20, 30, 40, 50])
mean = np.mean(data)

# Centrage
centered_data = data - mean
print(centered_data)

# Standardisation : moyenne 0 et écart type 1
std_dev = np.std(data)
standardized_data = (data - mean) / std_dev
print(standardized_data)

# Transformation logarithmique (valeurs strictement positives)
log_transformed_data = np.log(data)
print(log_transformed_data)
```

Python

```
[-20. -10.   0.  10.  20.]
[-1.41421356 -0.70710678  0.          0.70710678  1.41421356]
[2.30258509 2.99573227 3.40119738 3.68887945 3.91202301]
```

L'échantillonnage aléatoire

L'échantillonnage aléatoire consiste à sélectionner un sous-ensemble de points dans un jeu de données plus large. Pour obtenir des résultats reproductibles, crée un générateur avec une graine (`seed`) grâce à `np.random.default_rng()` :

- **Échantillonnage aléatoire simple :** on sélectionne un échantillon aléatoire d'une taille donnée. Sans remise (`replace=False`), chaque élément sélectionné n'est pas remis dans la population ;
- **Échantillonnage bootstrap :** on échantillonne avec remise (`replace=True`) pour créer plusieurs jeux de données. Cette méthode sert souvent à estimer la variabilité d'une statistique.

```python
rng = np.random.default_rng(seed=42)
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Échantillonnage aléatoire simple, sans remise
random_samples = rng.choice(data, size=5, replace=False)
print(random_samples)

# Échantillonnage bootstrap : 1000 échantillons de la taille du jeu de données
num_samples = 1000
bootstrap_samples = rng.choice(data, size=(num_samples, len(data)), replace=True)
print(bootstrap_samples.shape)

# Variabilité de la moyenne estimée par bootstrap
print(bootstrap_samples.mean(axis=1).std())
```

Python

```
[6 4 8 1 5]
(1000, 10)
0.9100573828061613
```

Générer des nombres aléatoires

NumPy permet de générer des nombres aléatoires selon la distribution de ton choix :

- **Entiers :** `rng.integers()` génère des entiers dans une plage donnée ;
- **Distribution uniforme :** `rng.uniform()` ;
- **Distribution normale :** `rng.normal()` ;
- **Distribution binomiale :** `rng.binomial()` simule des expériences binomiales ;
- **Distribution de Poisson :** `rng.poisson()` modélise des événements rares.

```python
rng = np.random.default_rng(seed=0)

print(rng.integers(low=1, high=7, size=5))          # dés à 6 faces
print(rng.uniform(low=0, high=1, size=3))           # uniforme entre 0 et 1
print(rng.normal(loc=0, scale=1, size=3))           # normale (moyenne 0, écart type 1)
print(rng.binomial(n=10, p=0.5, size=5))            # 10 lancers de pièce, 5 fois
print(rng.poisson(lam=3, size=5))                   # 3 événements en moyenne
```

Python

```
[6 4 4 2 2]
[0.01652764 0.81327024 0.91275558]
[ 1.30400005  0.94708096 -0.70373524]
[7 6 1 7 2]
[4 1 3 6 3]
```

Les anciennes fonctions `np.random.choice()`, `np.random.randint()` ou `np.random.normal()` fonctionnent toujours, mais le générateur `default_rng()` est aujourd'hui l'approche recommandée par NumPy.

La suite de la série porte sur NumPy pour l'algèbre linéaire. Documentation : générateurs aléatoires de NumPy.

Pour aller plus loin, consulte aussi notre guide des 45 méthodes NumPy.

Moyenne et écart-type ne suffisent pas à comprendre un jeu de données. Le cours Analyse et visualisation de données t'apprend à explorer, résumer puis visualiser des données réelles.


NUMPY