

# Logiciels scientifiques - HLMA310

## Partie 1 : Python

### Statistiques descriptives et pandas

**Joseph Salmon**

<http://josephsalmon.eu>

Université de Montpellier



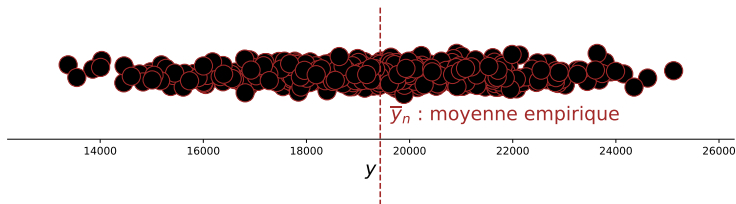
# Introduction

Base de données : `belgianmunicipalities.csv`, données des revenus des foyers belges, par région et par villes

```
from download import download
url = "http://josephsalmon.eu/enseignement/datasets/belgianmunicipalities.csv"
path_target = "./belgianmunicipalities.csv"
download(url, path_target, replace=False)
```

On s'intéressera ici qu'à la variable `medianincome` dans la suite.

# Moyenne (arithmétique)



**Définition : Moyenne (arithmétique)**

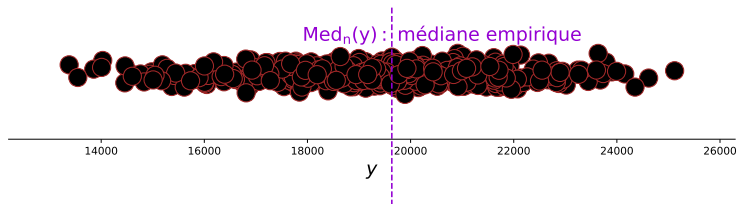
$$\bar{y}_n = \frac{1}{n} \sum_{i=1}^n y_i$$

Si  $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$  (produit scalaire) et  $\mathbf{1}_n = (1, \dots, 1)^\top \in \mathbb{R}^n$  :

$$\bar{y}_n = \left\langle \mathbf{y}, \frac{\mathbf{1}_n}{n} \right\rangle$$

**Exo:** Le vecteur  $\bar{y}_n \mathbf{1}_n$  est la projection de  $\mathbf{y}$  sur l'espace vect( $\mathbf{1}_n$ )

# Médiane



On ordonne les  $y_i$  dans l'ordre croissant :  $y_{(1)} \leq y_{(2)} \leq \dots \leq y_{(n)}$

---

---

**Définition : Médiane**

---

---

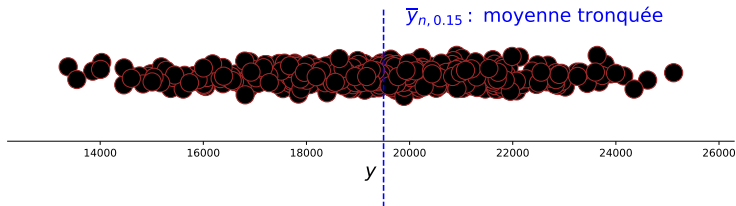
$$\text{Med}_n(\mathbf{y}) = \begin{cases} \frac{y_{(\frac{n}{2})} + y_{(\frac{n}{2}+1)}}{2}, & \text{si } n \text{ est pair} \\ y_{(\frac{n+1}{2})}, & \text{si } n \text{ est impair} \end{cases}$$

---

---

Rem: la définition d'une médiane est non-unique, et peut être parfois ambigu...

# Moyenne tronquée



Pour un paramètre  $\alpha$  (e.g.,  $\alpha = 15\%$ ), on calcule la moyenne en enlevant les  $\alpha\%$  plus grandes et plus petites valeurs

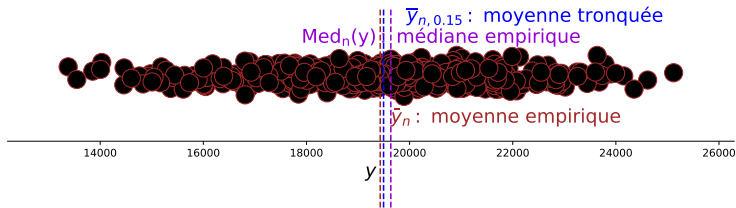
**Définition : Moyenne tronquée (à l'ordre  $\alpha$ )**


$$: \bar{y}_{n,\alpha} = \bar{z}_n$$

où  $\mathbf{z} = (y_{(\lfloor \alpha n \rfloor)}, \dots, y_{(\lfloor (1-\alpha)n \rfloor)})$  est l'échantillon  $\alpha$ -tronqué

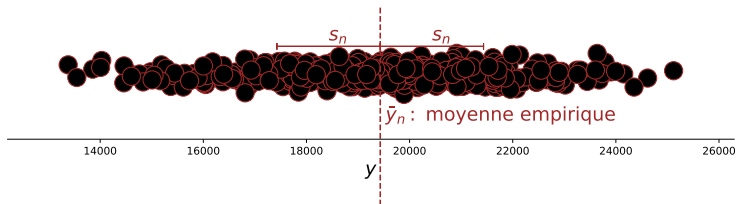
Rem:  $\lfloor u \rfloor$  est le nombre entier tel que  $\lfloor u \rfloor - 1 < u \leq \lfloor u \rfloor$

# Moyenne vs médiane



- ▶ Les trois statistiques ne coïncident pas
- ▶ Moyennes tronquées et médianes sont robustes aux points atypiques ( : *outliers*), la moyenne non !

# Dispersion : variance / écart-type



---

---

## Définitions

---

---

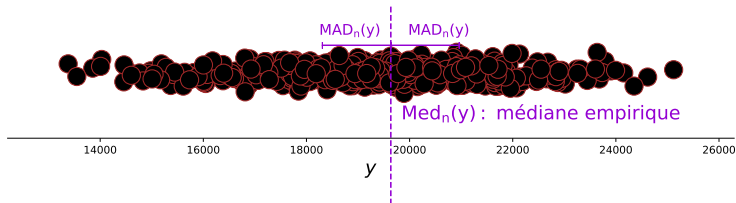
**Variance :** 
$$\text{var}_n(\mathbf{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}_n)^2 = \frac{1}{n} \|\mathbf{y} - \bar{y}_n \mathbf{1}_n\|^2$$

**Écart-type :** 
$$s_n(\mathbf{y}) = \sqrt{\text{var}_n(\mathbf{y})} \quad (\text{où } \|\mathbf{z}\|^2 = \sum_{i=1}^n z_i^2)$$

---

---

# Dispersion : MAD



---

---

## Définition

---

---

**Déviati n m diane absolue** ( : *Mean Absolute Deviation*) :

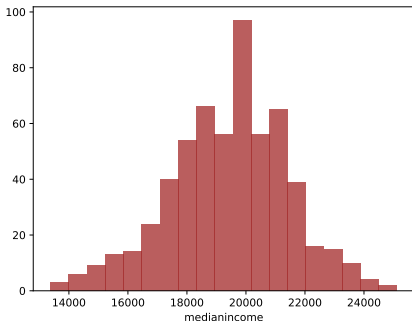
$$\text{MAD}_n(\mathbf{y}) = \text{Med}_n (|\text{Med}_n(\mathbf{y}) - \mathbf{y}|)$$

---

---



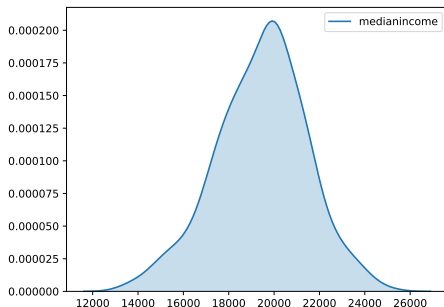
# Estimation de la densité : histogramme



L'**histogramme** est une approximation de la **densité** par une fonction constante par morceaux

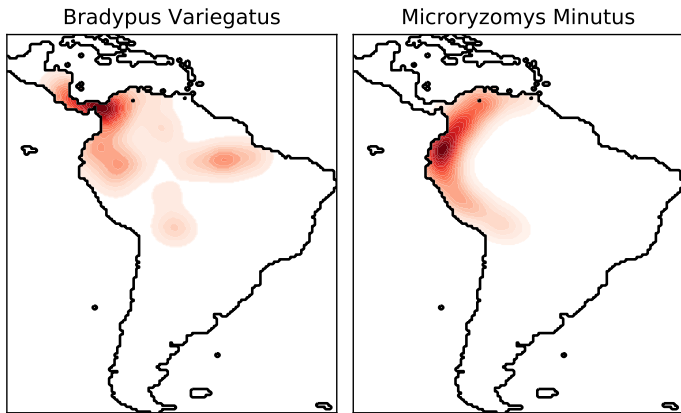
Rem: les « cases » (🇬🇧 : *bins*) ont une aire proportionnelle au nombre de données qu'elles contiennent

# Estimation de la densité : méthode à noyau



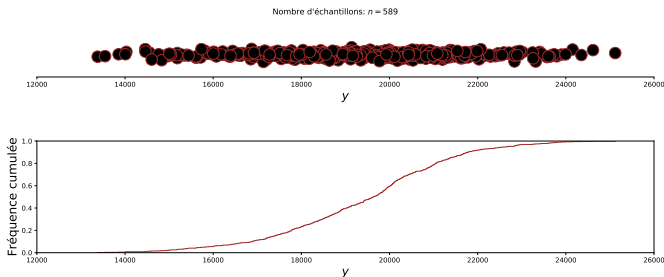
- Méthode à noyau (🇬🇧 : *Kernel Density Estimation, KDE*) :  
approche non-paramétrique estimant la densité par une  
fonction continue – généralisation de l'histogramme

# Densité bi-dimensionnelle (spatiale)



cf. [http://scikit-learn.org/stable/\\_downloads/plot\\_species\\_kde.py](http://scikit-learn.org/stable/_downloads/plot_species_kde.py)

# Fonction de répartition



---

---

**Définition : fonction de répartition**

---

---

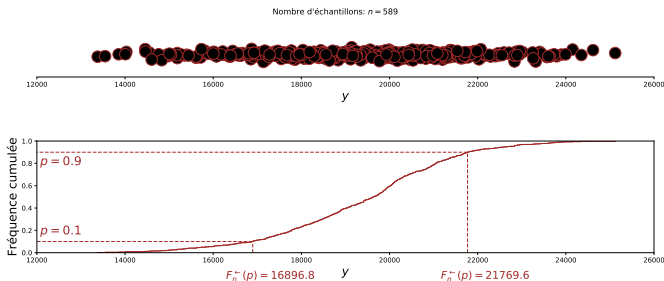
**Empirique :** 
$$F_n(u) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{\{y_i \leq u\}}$$

---

---

Interprétation : proportion d'observations sous un certain niveau

# Fonction quantile



---

**Définition : fonction quantile**

---

Pour  $p \in ]0, 1]$ ,  $F_n^{\leftarrow}(p) = \inf\{u \in \mathbb{R} : F_n(u) \geq p\}$

---

Rem: c'est l'inverse (généralisée) de la fonction de répartition ; sa définition admet plusieurs conventions, cf. **percentile** in `numpy`