

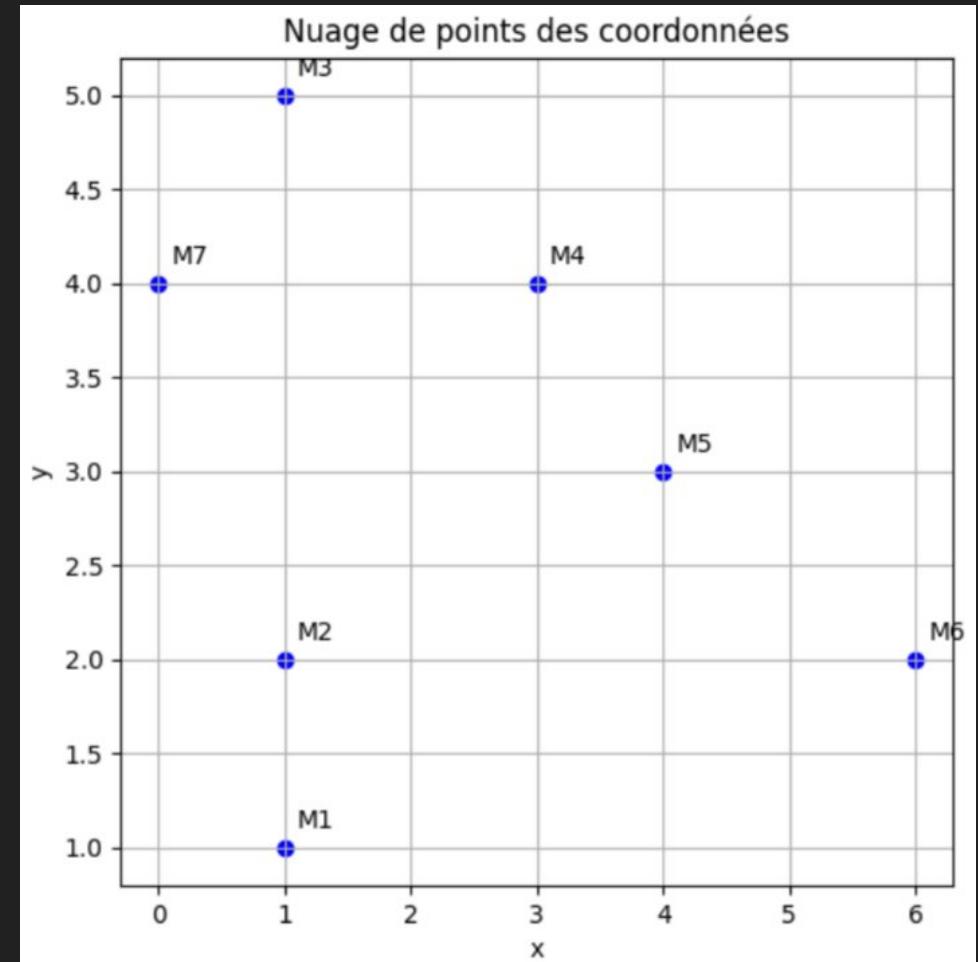
Projet Électif

Noé LORRET-DESPRET & Clément AUVRAY

Statistiques Descriptives

	x	y
M1	1	1
M2	1	2
M3	1	5
M4	3	4
M5	4	3
M6	6	2
M7	0	4

	x	y
Moyenne	2.286	3.000
Médiane	1.000	3.000
Variance	4.571	2.000
Écart-type	2.138	1.414
Min	0.000	1.000
Max	6.000	5.000
Étendue	6.000	4.000



Régression Linéaire Simple

Coefficients de régression

$$b_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$

Somme des carrés

$$SCE = \sum_{i=1}^n (y_i - \hat{y})^2$$

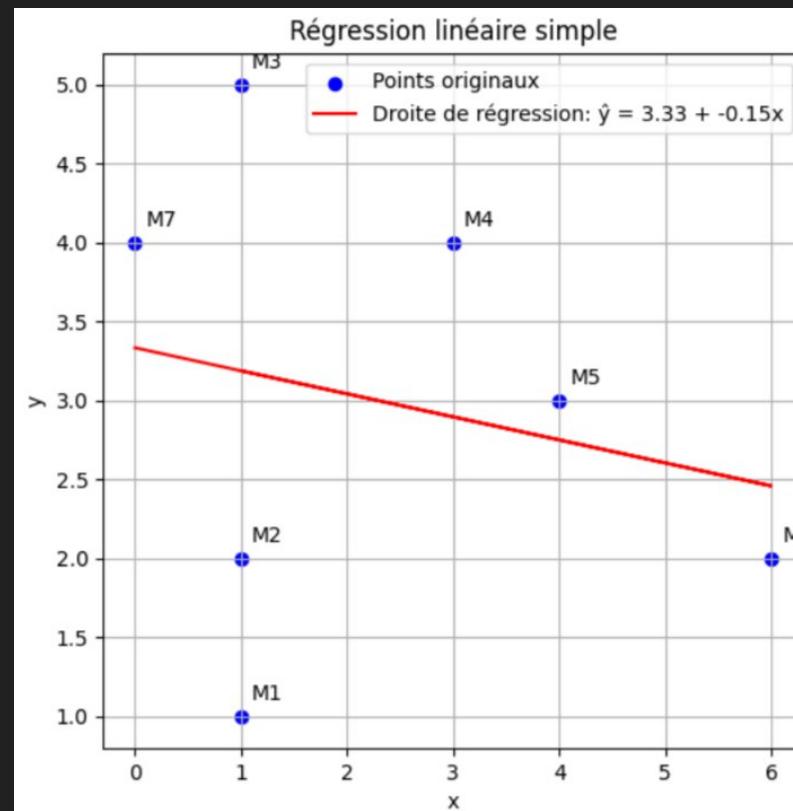
$$SCT = \sum_{i=1}^n (y_i - \bar{y})^2$$

$$SCR = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

$$SCT = 12.0000$$

$$SCE = 11.4167$$

$$SCR = 0.5833$$



Coefficient de détermination

$$R^2 = 1 - \frac{SCE}{SCT} = \frac{SCR}{SCT}$$

$$R^2 = 0.0486$$

Estimation de l'erreur

Variance estimée des erreurs

$$\sigma^2 = MSE = \frac{SCE}{n - 2}$$

Écart-type des erreurs

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{SCE}{n - 2}}$$

Pourquoi $n - 2$?

MSE : 2.2833

Écart-type des erreurs : 1.5111

Tests statistiques

Erreur standard des coefficients

$$SE_{b_1} = \frac{\sigma}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

$$SE_{b_0} = \sigma \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

Erreur standard de b1 : 0.2885
Erreur standard de b0 : 0.8724

Tests d'hypothèse

$$t = \frac{b_i - 0}{SE_{b_i}}$$

t statistique pour b1 : -0.5054
t statistique pour b0 : 3.8208
Valeur critique t (alpha=0.05, ddl=5) : 2.5706

Intervalles de confiance

$$\beta \pm t_{\alpha/2, n-2} \times SE_{\beta}$$

Intervalle de confiance à 95% pour b1 : [-0.8875, 0.5958]
Intervalle de confiance à 95% pour b0 : [1.0907, 5.576]

Classification Ascendante Hiérarchique

Explication de l'algorithme

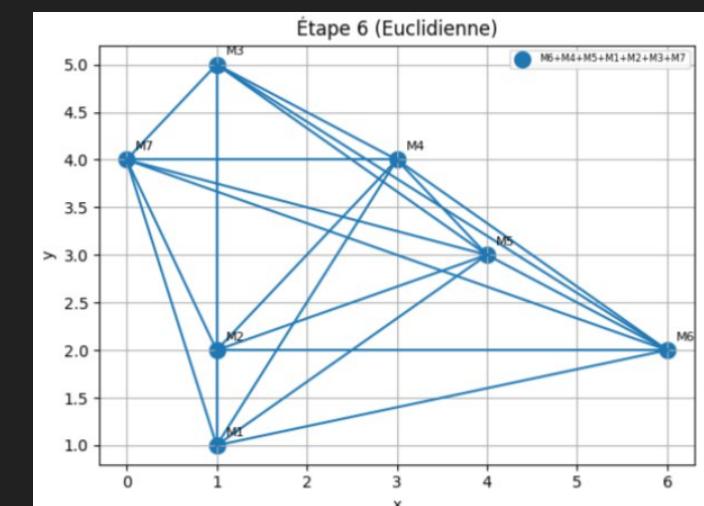
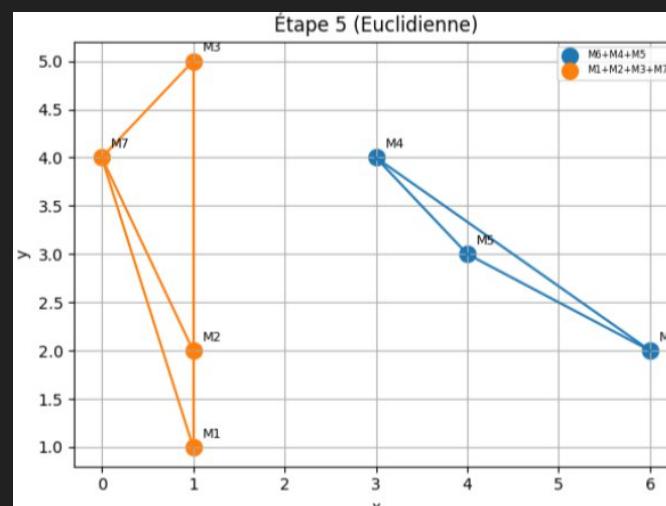
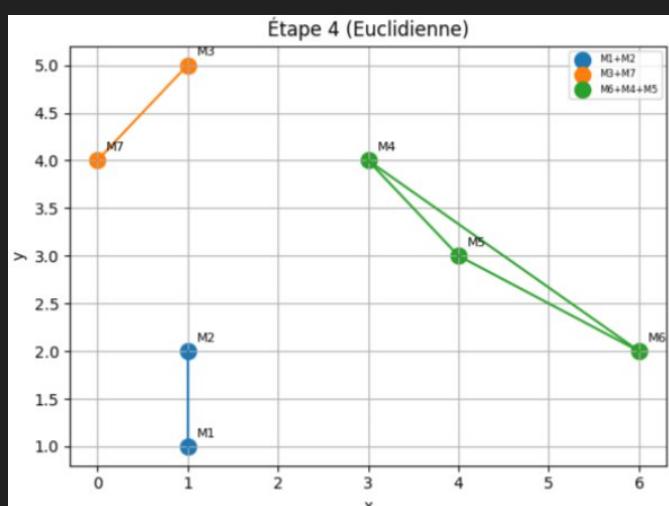
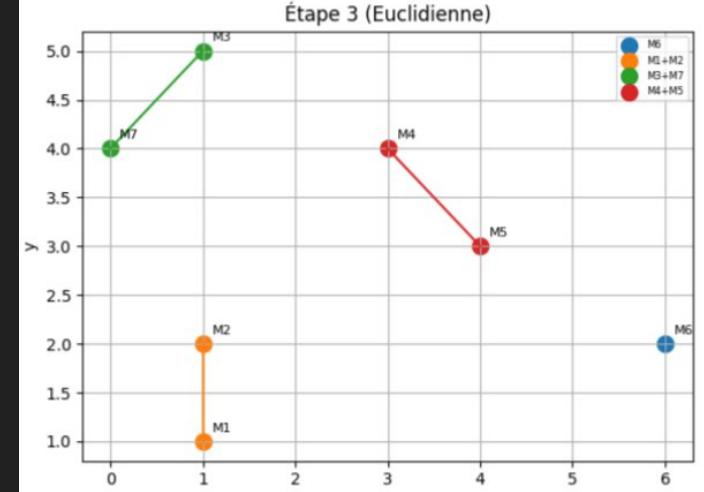
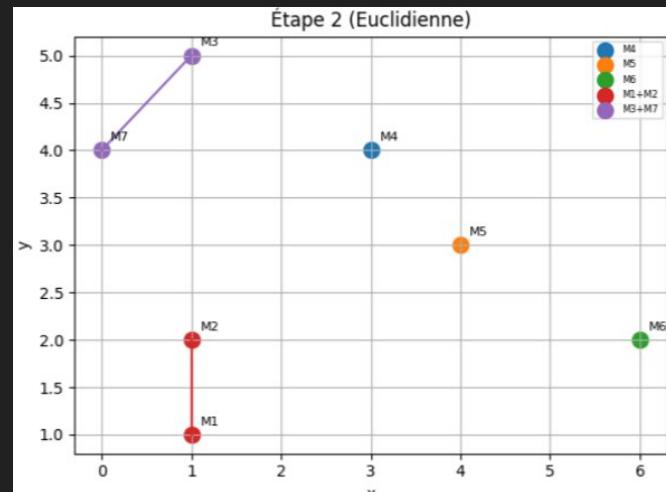
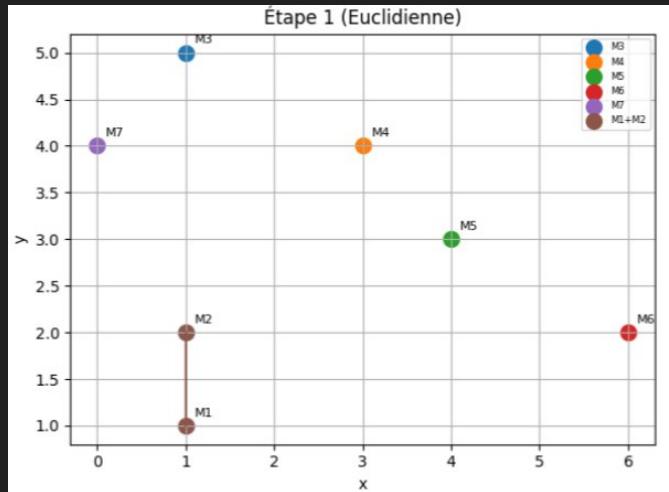
```
def dist_euclidienne(p1: pd.Series, p2: pd.Series) -> float:  
    '''Distance euclidienne entre 2 points'''  
  
    return sqrt((p1['x'] - p2['x']) ** 2 + (p1['y'] - p2['y']) ** 2)
```

```
def dist_group(g1: str, g2: str, clusters: dict, df: pd.DataFrame, dist_func: Callable) -> float:  
    '''Distance minimale entre groupes'''  
  
    return min(dist_func(df.loc[p1], df.loc[p2]) for p1 in clusters[g1] for p2 in clusters[g2])
```

```
def dist_min(t: pd.DataFrame, clusters: dict, dist_func: Callable) -> tuple:  
    '''Retourne la paire de clusters les plus proches selon une fonction de distance donnée.'''  
  
    # Liste des clés (identifiants des clusters)  
    keys = list(clusters.keys())  
  
    # Génère toutes les combinaisons uniques ( $i < j$ ) d'indices de clusters  
    combi = np.vstack(np.triu_indices(len(keys), k=1))[:, 0]  
  
    # Calcule les distances entre chaque paire de clusters avec la fonction fournie  
    combi_dist = [  
        dist_group(keys[i], keys[j], clusters, t, dist_func)  
        for i, j in combi  
    ]  
  
    # Trouve l'indice de la paire avec la plus petite distance  
    i, j = combi[np.argmin(combi_dist)]  
    closest_pair = (keys[i], keys[j])  
    return closest_pair
```

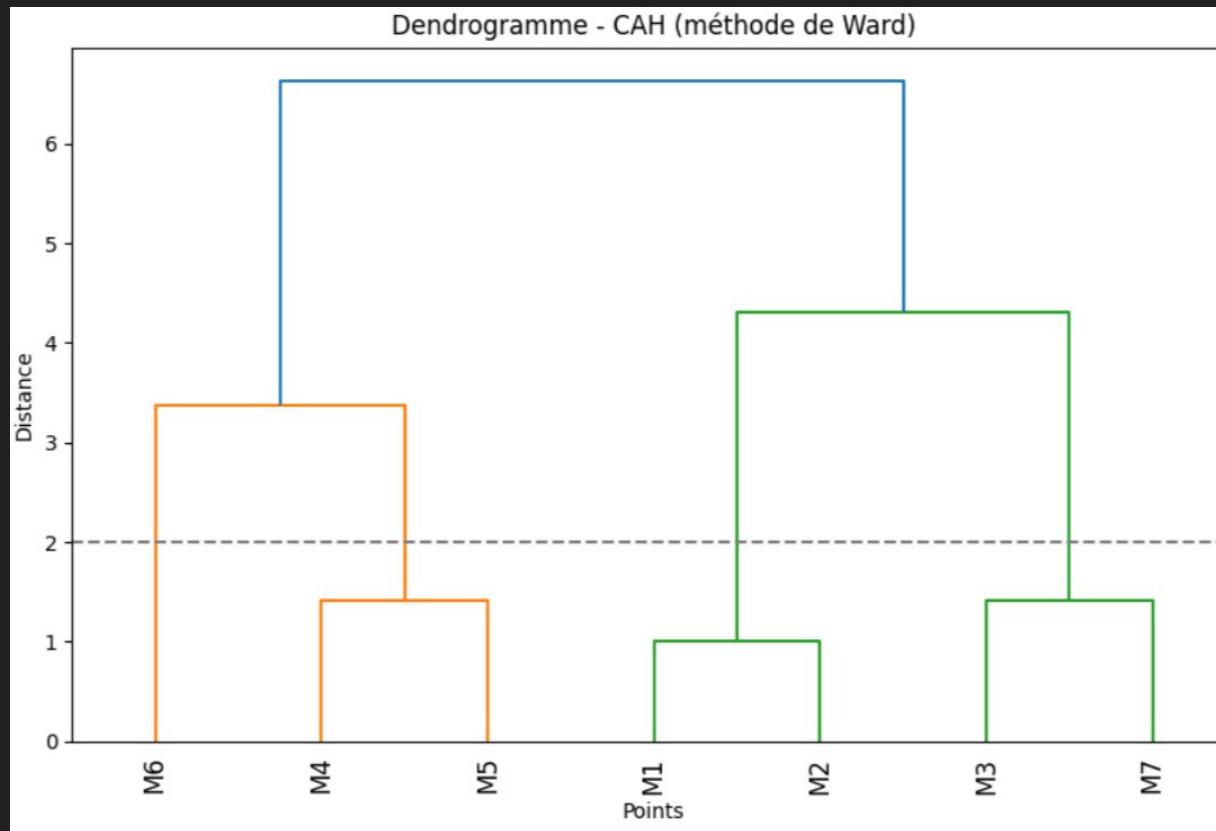
Classification Ascendante Hiérarchique

Application et résultats



Classification Ascendante Hiérarchique

Application et résultats



Évaluation et validation du clustering

Indices d'évaluation

Silhouette Score (29, 9) : 0.4006

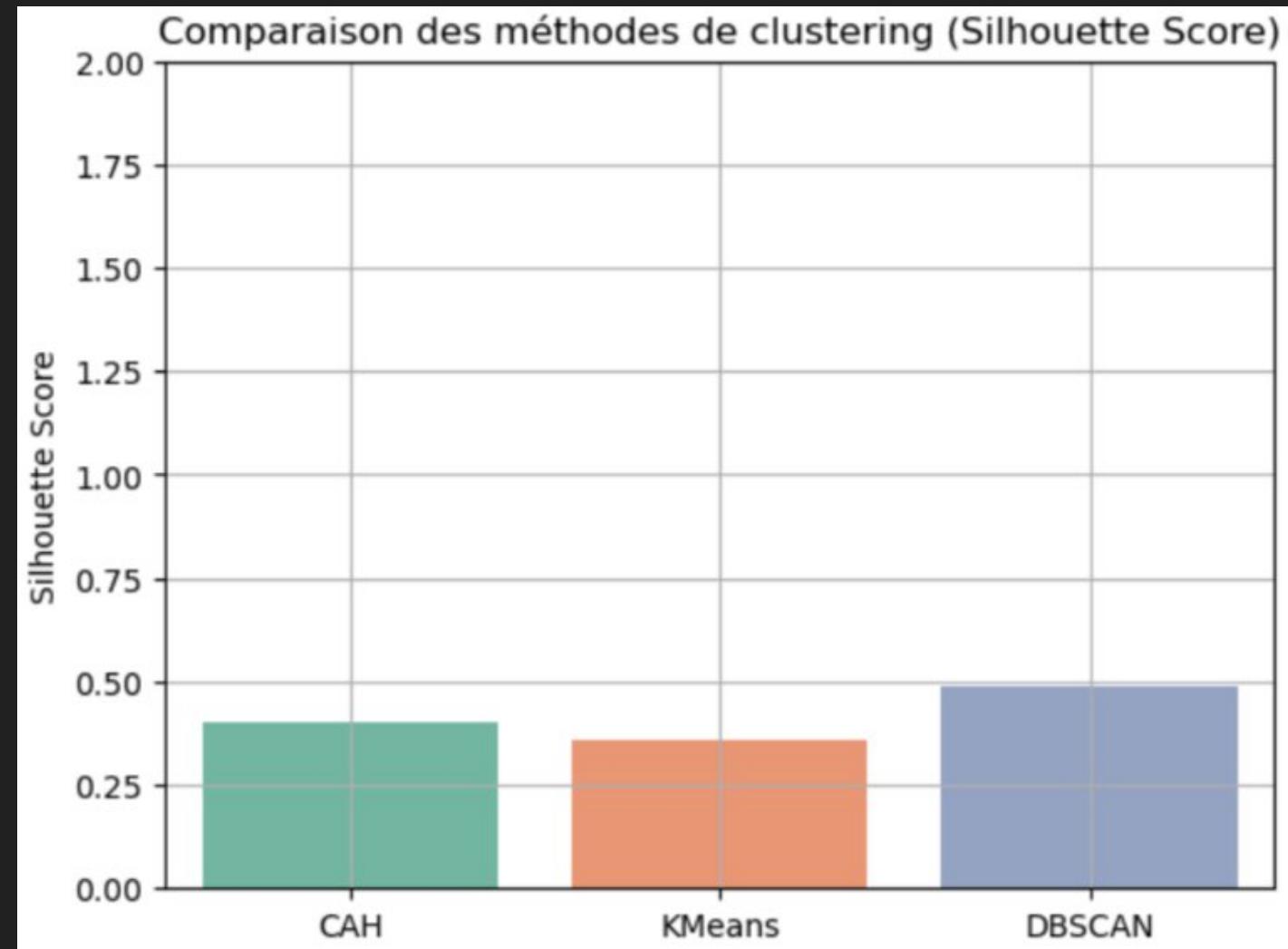
Indice de Dunn : 0.4756

Cohésion intra-cluster : 51.2785
Séparation inter-cluster : 14.6867

Évaluation et validation du clustering

Validation croisée

```
Silhouette Score CAH      : 0.4006
Silhouette Score KMeans   : 0.3603
Silhouette Score DBSCAN   : 0.4907
ARI CAH - KMeans         : 0.5924
ARI CAH - DBSCAN         : 0.5656
ARI DBSCAN - Kmeans      : 0.3811
```



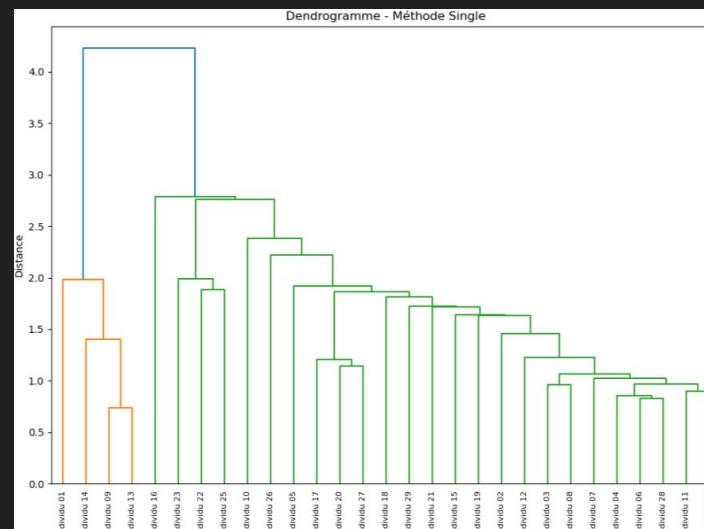
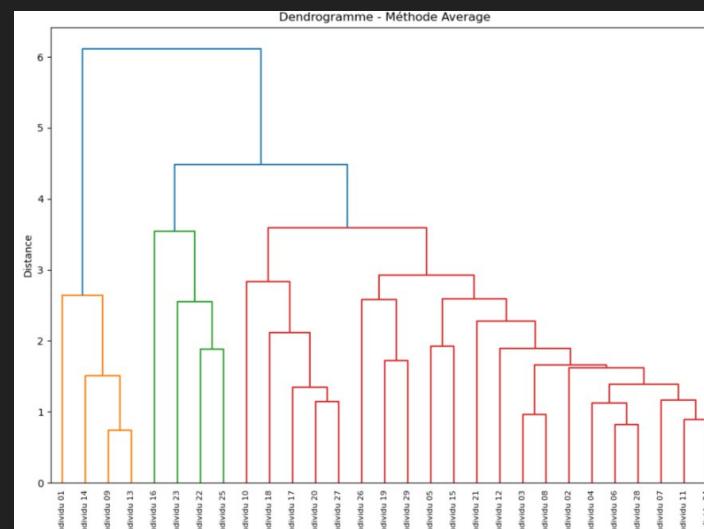
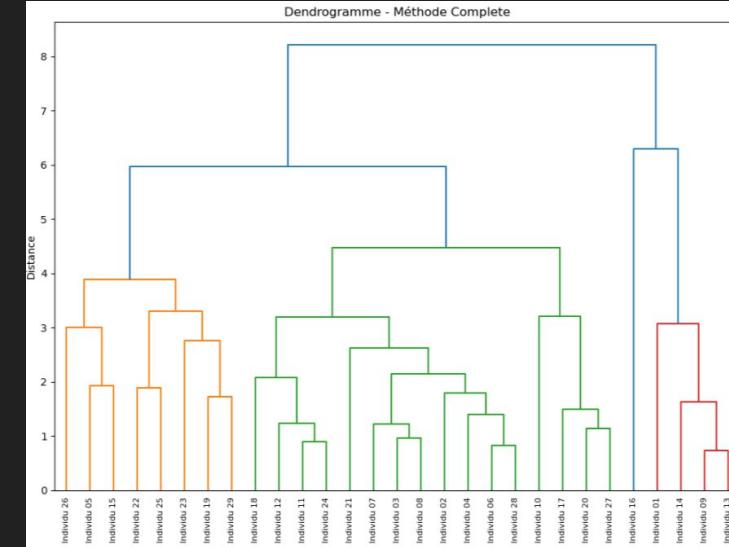
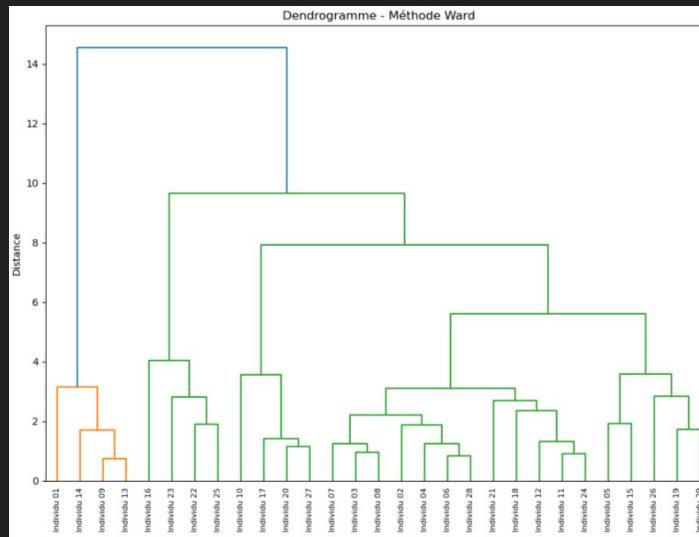
Évaluation et validation du clustering

Interprétation des résultats

	Données 1	Données 2	Données 3	Données 4	Données 5	Données 6	Données 7	Données 8	Données 9
1 - mean	101.750	44.750	133.750	6.275	55.150	16.475	7.200	18.250	11.250
1 - median	97.500	33.000	120.550	5.650	56.700	20.200	7.650	16.500	10.500
1 - std	33.049	31.941	61.850	3.431	10.462	9.122	2.288	8.884	1.893
2 - mean	276.500	235.500	127.200	22.825	115.000	34.050	17.950	65.000	24.250
2 - median	278.000	234.000	110.050	22.450	109.700	34.400	18.650	65.000	22.500
2 - std	57.512	71.636	64.445	4.551	25.989	2.726	5.133	12.910	8.655
3 - mean	342.286	236.738	206.786	27.819	60.890	8.343	23.062	87.143	30.476
3 - median	338.000	238.000	211.100	28.000	57.600	5.700	22.400	80.000	28.000
3 - std	36.318	96.284	66.909	3.044	13.117	8.063	4.435	16.169	10.186

Évaluation et validation du clustering

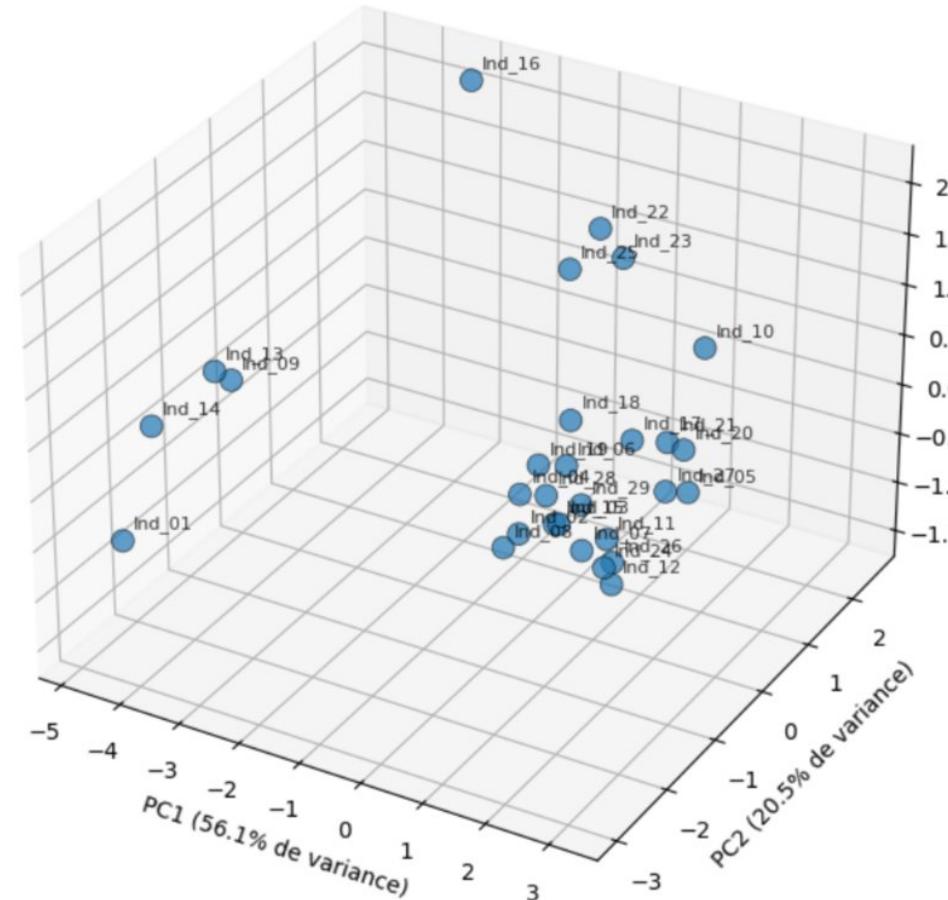
Analyse critique de la méthode



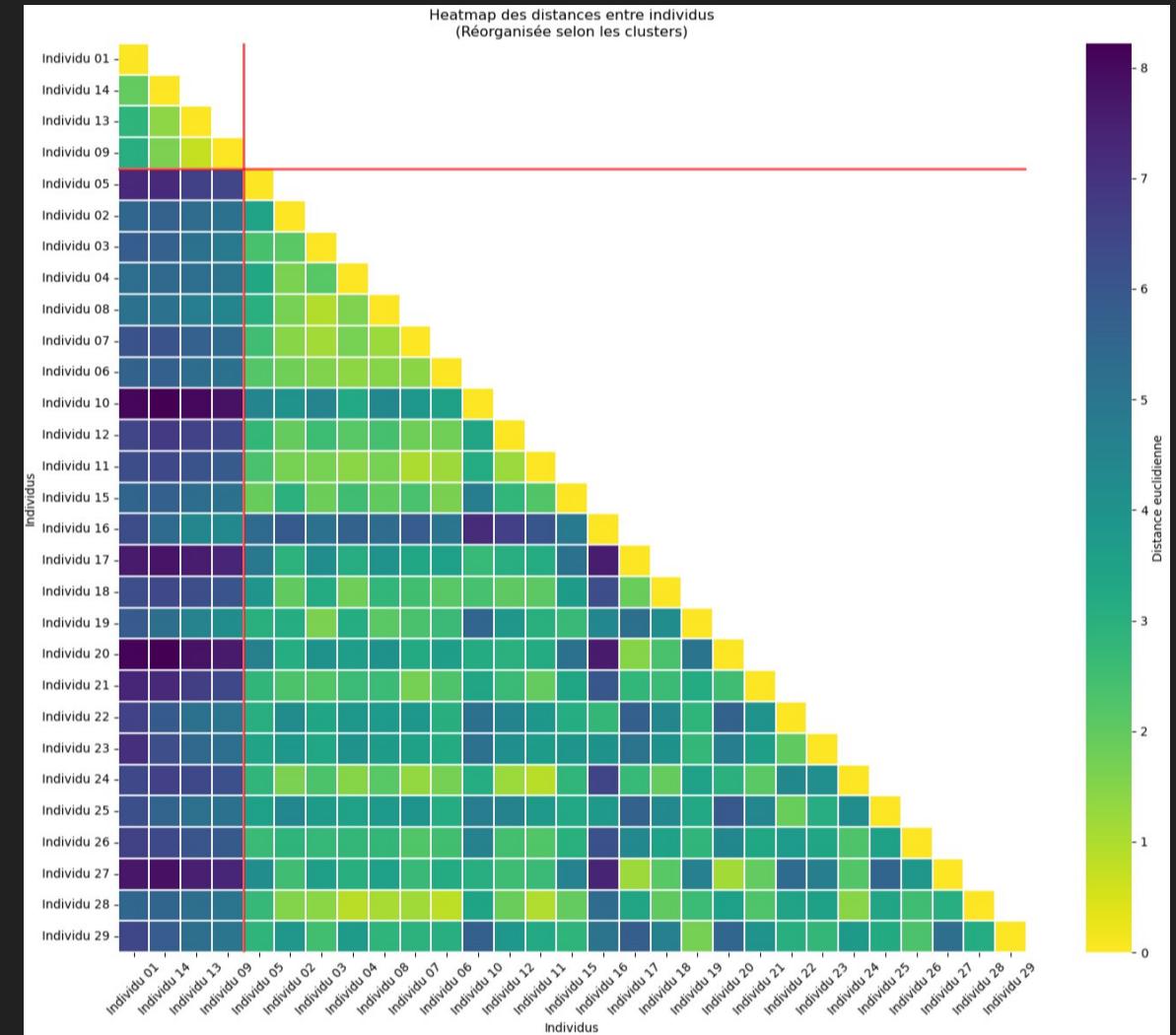
Évaluation et validation du clustering

Visualisation avancée

Projection ACP - Plan des deux premières composantes principales



Heatmap des distances entre individus
(Réorganisée selon les clusters)



Merci de votre écoute



h1 title

h2 title

h3 title

p text

p text

p value

p value

p value

p info

p warning

p good

```
ts convert.ts l@0.2.x theme.lua util.lua colors.lua package.ts aiohttp .eslintrc.js ts[+]
import fs from 'fs';
import path from 'path';
import util from 'util';
function err() {} // 'util' is declared but its value is never read.
err();
// prettier/prettier : Delete ``
interface FindPackagesOption {
  includeRoot?: boolean;
  ignore?: string[];
  cwd?: string;
}

const BASIC_IGNORE = ['**/test/**', '**/tests/**', '**/_tests_/**'];
const DEFAULT_IGNORE = ['**/test/**', '**/tests/**', '**/_tests_/**'];

export async function findPackages(
  patterns: string[],
  options?: FindPackagesOption
) {
  const fastGlob = (await import('fast-glob')).default;
  if (!options) options = {};
  if (options.ignore) options.ignore = DEFAULT_IGNORE;
  options.ignore.push(...BASIC_IGNORE);
  if (options.includeRoot) patterns.push('*');
  patterns = patterns.map(pattern => {
    pattern.replace(/\//g, '/');
  });
  return (await fastGlob(patterns, options)).map((file) =>
    path.resolve(options.cwd || process.cwd(), path.dirname(file))
  );
}

NORMAL 3 masters ① 9 △ 1 ② 1 > package.ts[+]
```

JetBrainsMono
MonaspaceKr

main.py

```
if __name__ == '__main__':
    httpd = HTTPServer((DOMAIN, PORT), Handler)
    httpd.serve_forever()
```