

Rapport

Analyse de données pour guider les choix d'affectation de l'aide humanitaire internationale

1. Objectifs du projet

L'objectif de ce projet est de réaliser une étude globale sur les différents pays du monde en matière de développement, afin de mettre en place une affectation de l'aide humanitaire internationale de manière optimale. Le but est donc de déterminer quels pays ont besoin en priorité de cette aide, en se basant sur des critères socio-économiques et de santé publique. Le projet se basera sur l'étude de différentes méthodes de clustering qui, une fois croisées, permettront d'établir une liste juste et représentative des pays dans le besoin.

Plus concrètement, ce rapport est composé d'une étude des différentes méthodes de clustering employées, puis d'une analyse de la liste de pays obtenus pour chacune d'elles, et est conclu par une liste globale faisant état des résultats croisés des méthodes utilisées.

2. Etude préalable

Cette partie est consacrée à la présentation des 4 algorithmes de partitionnement qui seront utilisés pour l'étude, à savoir :

- La méthode DBSCAN
- La méthode des K-Means
- La classification hiérarchique ascendante
- Le modèle de mélange des Gaussiennes

Tout particulièrement, est effectué une présentation de la méthode DBSCAN, qui n'a pas été vue en cours ni en TP.

Partie 2.1 : METHODE DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*)

Le DBSCAN est un algorithme de partitionnement qui s'appuie sur des considérations de densité des données pour établir les clusters. Pour se faire, l'algorithme prend en compte deux attributs : une distance **Epsilon** et un nombre de points **MinPts** devant se trouver dans le rayon **Epsilon** de la donnée visitée.

Fonctionnement de l'algorithme DBSCAN :

Initialisation : On sélectionne une donnée aléatoire et on vérifie qu'elle possède au moins **MinPts** voisins dans un rayon **Epsilon**. Si tel est le cas, ce point devient le premier d'un nouveau cluster. Sinon, il est considéré (temporairement) comme du bruit.

Itérations : Les points du voisinage de la donnée précédente sont inclus dans le cluster. On répète cette étape tant qu'il reste des points du voisinage du cluster non visités. Une fois le cluster complété, on recommence à l'initialisation en partant d'un point non visité.

En fin d'itérations, on obtient un partitionnement de l'espace en K clusters et une élimination de données aberrantes.

Cet algorithme possède plusieurs avantages pour le clustering, mais aussi des inconvénients :

AVANTAGES	INCONVENIENTS
Il est efficace pour éliminer les points aberrants : il élimine automatiquement du partitionnement les points qui n'ont pas un voisinage assez fourni.	Les paramètres Epsilon et MinPts sont difficiles à évaluer par l'utilisateur.
L'algorithme est assez simple : il n'est ni plus ni moins qu'un parcours de graphe. Le DBSCAN détermine lui-même le nombre de clusters.	L'implémentation de l'algorithme est peu judicieuse pour des clusters avec des densités différentes.
Il permet de déterminer des clusters non ronds, comme des anneaux. Cela peut s'avérer utile pour mettre en valeur des similarités comportementales des données.	Le DBSCAN ne fonctionne pas bien lorsque la dimension du jeu de données est trop grande. En effet, plus la dimension augmente et plus les données s'éloignent dans l'espace.

Partie 2.2 : PRESENTATION DES METHODES

Les méthodes de clustering utilisent toutes des données centrées réduites. Ainsi, les jeux de données fournis doivent être traités au préalable.

Pour se faire, on utilise la classe `StandardScaler` du module `sklearn` :

```
X = StandardScaler().fit_transform(Data)
```

Ici, **X** est le jeu de données **Data** ayant été centré et réduit.

METHODE K-MEANS : classe `sklearn.cluster.KMeans`

Cette classe effectue l'algorithme K-Means.

Paramètres : (liste non exhaustive)

- **n_clusters** : Nombre de clusters souhaités (seul paramètre qui ne peut être défini par défaut)
- **init** : Initialisation des centres de cluster initiaux de façon aléatoire à partir des centres de gravités initiaux (random) ou de façon intelligente (k-means++)
- **n_init** : Nombre de fois où on exécute l'algorithme K-Means avec des centroïdes différents (l'algorithme retourne la meilleure exécution en terme de répartition d'inertie entre les clusters)
- **max_iter** : Nombre maximum d'itérations de l'algorithme pour une exécution. Si on ne le fixe pas, l'algorithme s'arrête lorsqu'il y a convergence

Attributs :

- **cluster_centers_** : Renvoie les coordonnées des clusters
- **labels_** : Indique à quel cluster appartient chaque donnée
- **inertia_** : Renvoie la somme des distances au carré des échantillons à leur centre de cluster le plus proche
- **n_iter_** : Indique le nombre d'itérations exécutées

Méthodes : (liste non exhaustive)

- **fit** : Réalise le clustering
- **fit_predict** : Calcule les centres des clusters et indique l'appartenance de chaque échantillon à son cluster
- **predict** : Prédit le cluster le plus proche auquel appartient un échantillon de données

CLASSIFICATION HIERARCHIQUE ASCENDANTE : module `scipy.cluster.hierarchy`

Ce module permet de visualiser et d'établir une classification hiérarchique ascendante.

Fonctions : (liste non exhaustive)

- **fcluster** : Renvoie la classification des données en les regroupant par bloc (au sein d'un vecteur) selon leur appartenance à chaque cluster
- **linkage** : Effectue la classification d'un ensemble de données
- **dendrogram** : Génère le dendrogramme associé à un échantillon de données

MODELE DE MELANGE DE GAUSSIENNES : classe `sklearn.mixture.GaussianMixture`

Cette classe permet d'estimer les paramètres d'une distribution de mélange gaussienne.

Paramètres : (liste non exhaustive)

- **n_components** : Nombre de clusters souhaités
- **covariance_type** : Décrit le type de paramètres de covariances à utiliser :
 - o *full* : chaque composant a sa matrice de covariance
 - o *tied* : toutes les composants ont la même matrice de covariance
 - o *diag* : chaque composant a sa matrice de covariance diagonale
 - o *spherical* : chaque composant a sa variance unique
- **max_iter** : Nombre d'itérations EM à effectuer ;
- **n_init** : Nombre d'exécutions à effectuer par l'algorithme (on garde la meilleure)

Attributs : (liste non exhaustive)

- **weights_** : Renvoie les poids de chaque composant
- **means_** : Renvoie la moyenne de chaque composant
- **covariances_** : Renvoie la covariance de chaque composant ;
- **n_iter_** : Nombre de pas utilisés par le meilleur ajustement de EM pour atteindre la convergence

Méthodes : (liste non exhaustive)

- **fit_predict** : Estime les paramètres et prédit l'appartenance aux clusters d'un échantillon de données

DBSCAN : classe `sklearn.cluster.DBSCAN`

Cette classe permet d'effectuer un clustering DBSCAN.

Paramètres : (liste non exhaustive)

- **eps** : Distance maximale entre deux échantillons pour qu'ils soient dans le même voisinage
- **min_samples** : Nombre minimal de points devant se trouver dans un certain rayon pour que ces points soient considérés comme appartenant à un même cluster

Attributs : (liste non exhaustive)

- **labels_** : Indique l'appartenance des points à leur cluster

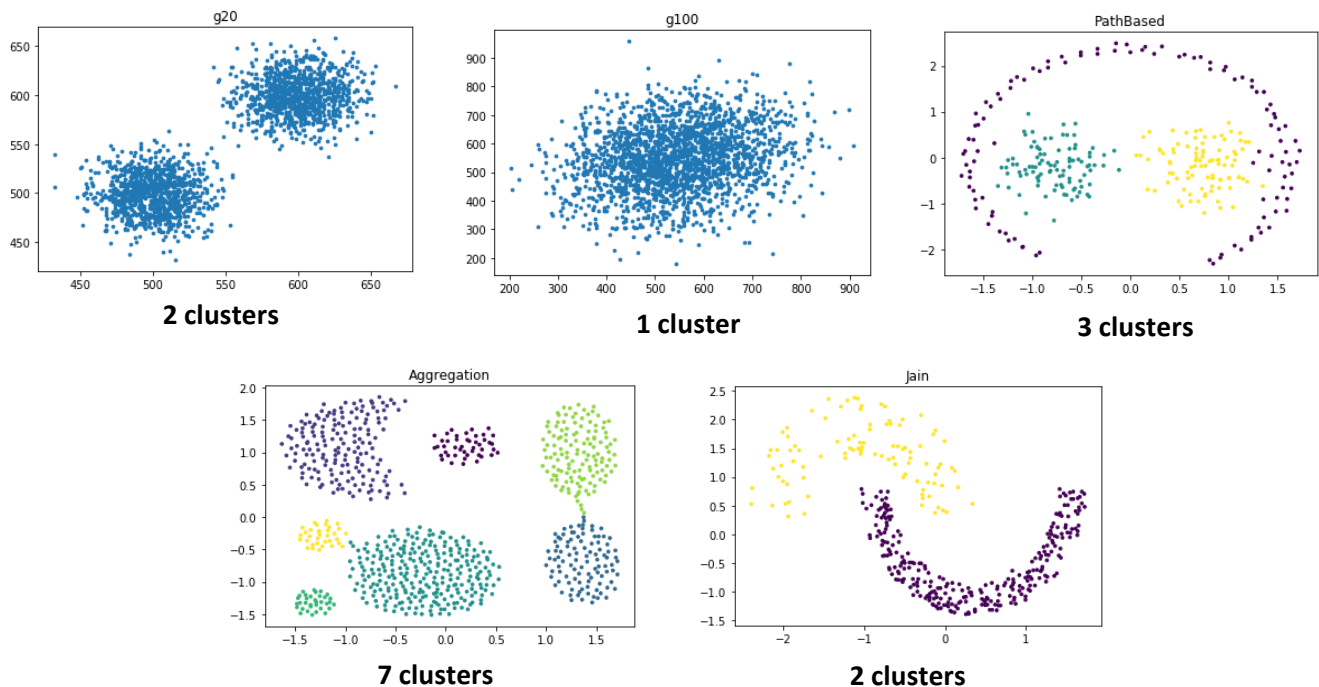
Méthodes : (liste non exhaustive)

- **fit_predict** : Effectue le clustering DBSCAN et renvoie l'appartenance des points à leur cluster

Partie 2.3 EXPERIMENTATIONS

Pour tester les 4 méthodes présentées, nous utilisons un jeu de données expérimentales regroupant des données agencées différemment dans l'espace. Cela permet de mettre à l'épreuve les différents algorithmes pour des traitements de données différents.

VISUALISATION DES NUAGES DE POINTS DES DONNEES TEST :



CHOIX DES PARAMETRES

Pour l'ensemble des méthodes, on décide de ne pas fixer de seuil maximal d'itérations, et de laisser l'algorithme converger naturellement.

K-Means : On fixe le nombre de clusters souhaités, une initialisation '*k-means++*' pour avoir une convergence la plus rapide possible, et on se contente d'une seule exécution, sauf dans le cas où l'on souhaite avoir une meilleure précision.

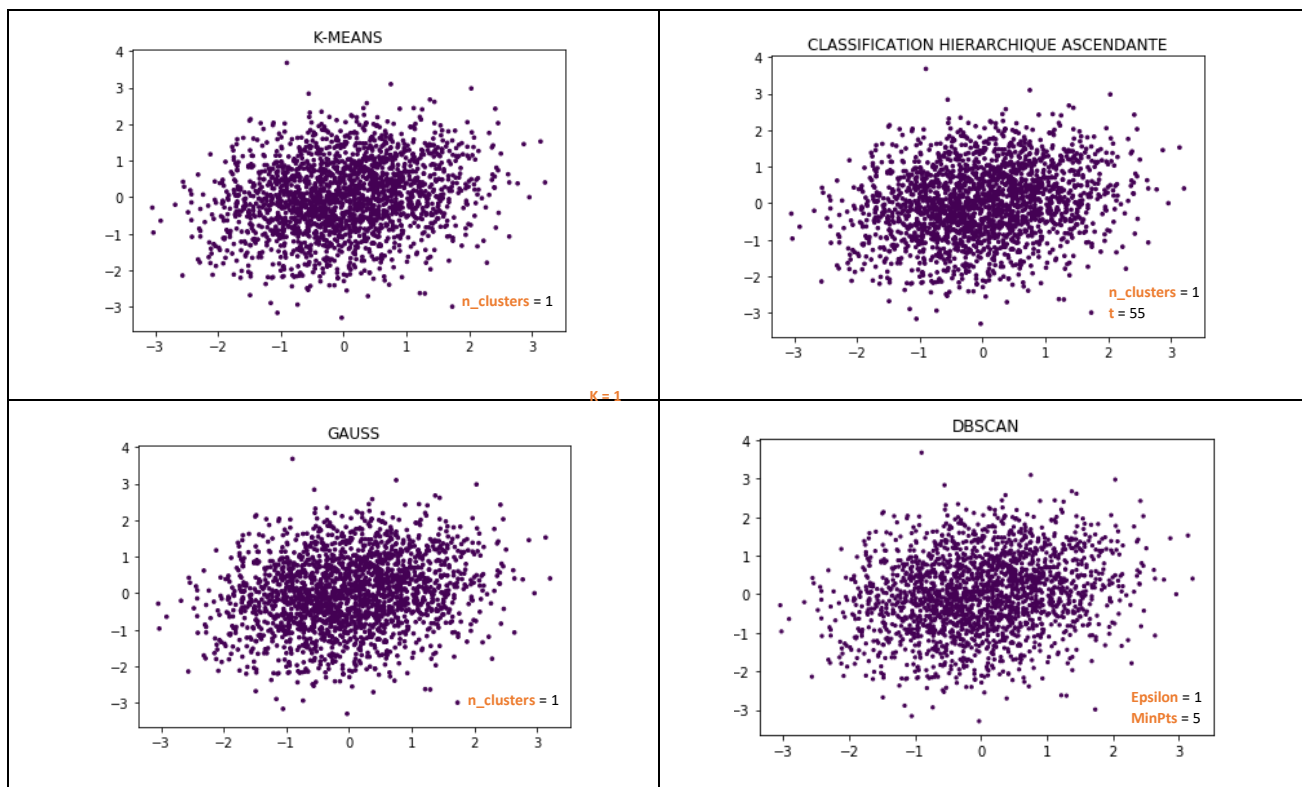
CAH : On applique la métrique '*Ward*' car c'est la plus optimale pour le clustering.

Gauss : On fixe le nombre de clusters souhaités. On effectue une seule initialisation (par défaut), sauf dans le cas où l'on souhaite avoir une meilleure précision.

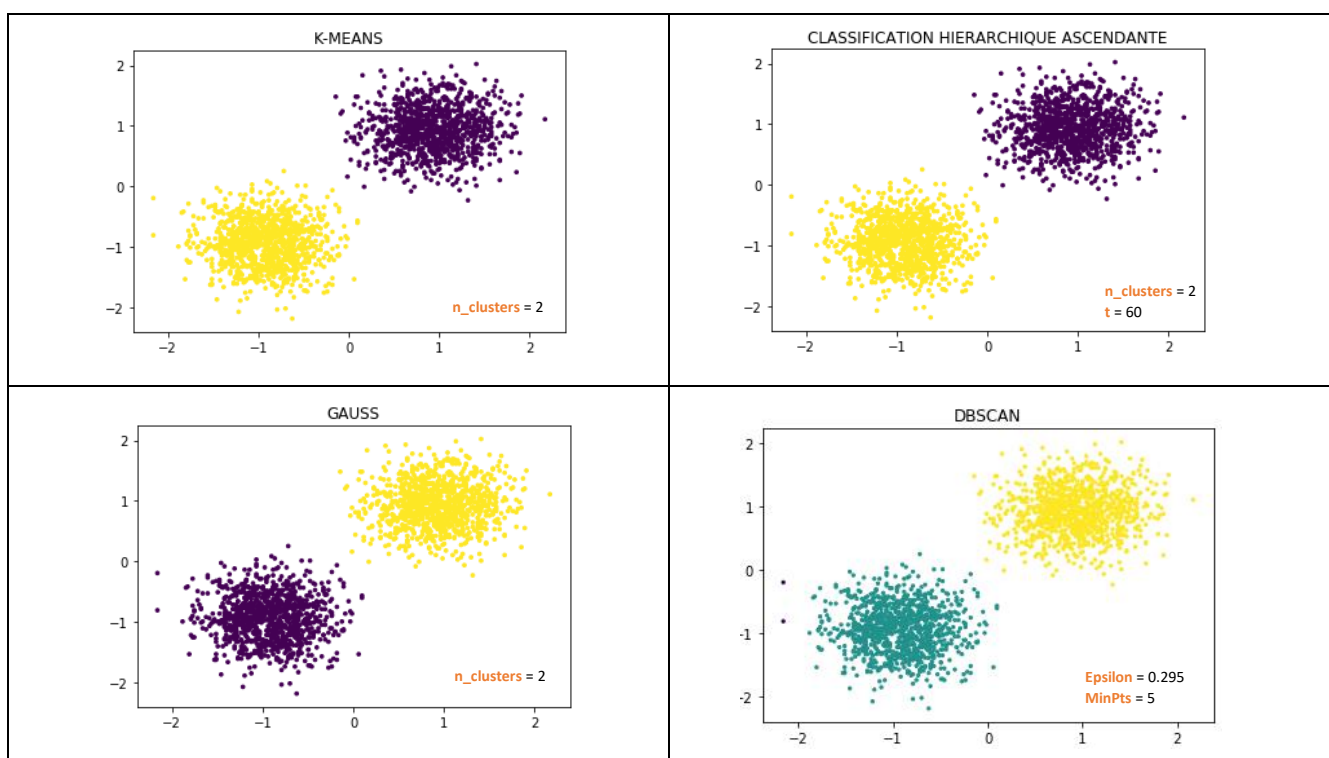
DBSCAN : Cette fois-ci, l'algorithme déduit lui-même le nombre de clusters. On lui indique cependant la distance maximale **eps** entre deux échantillons pour être **considérés** dans le même voisinage, et le nombre minimal de points **min_samples** devant se trouver dans un même voisinage pour être considérés comme appartenant à un même cluster.

ETUDE DES PARTITIONNEMENTS POUR CHAQUE JEU DE DONNEES

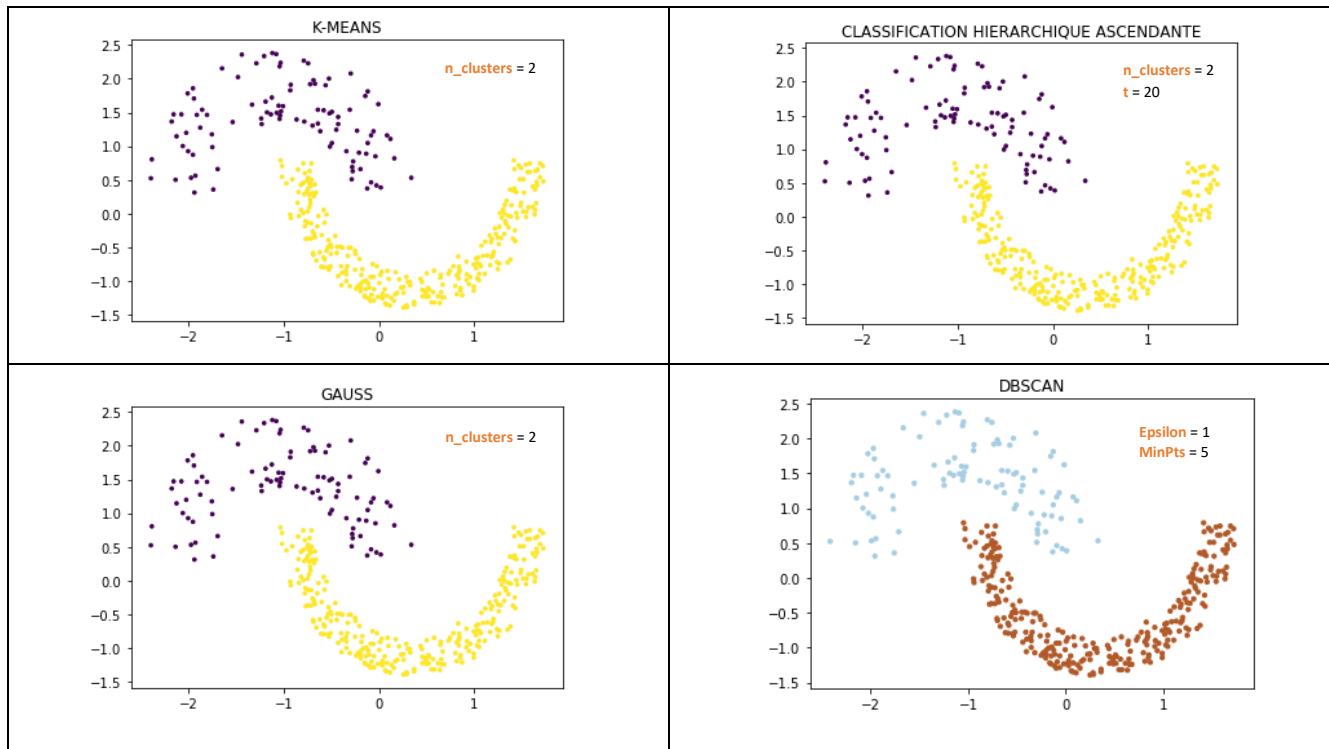
• G100



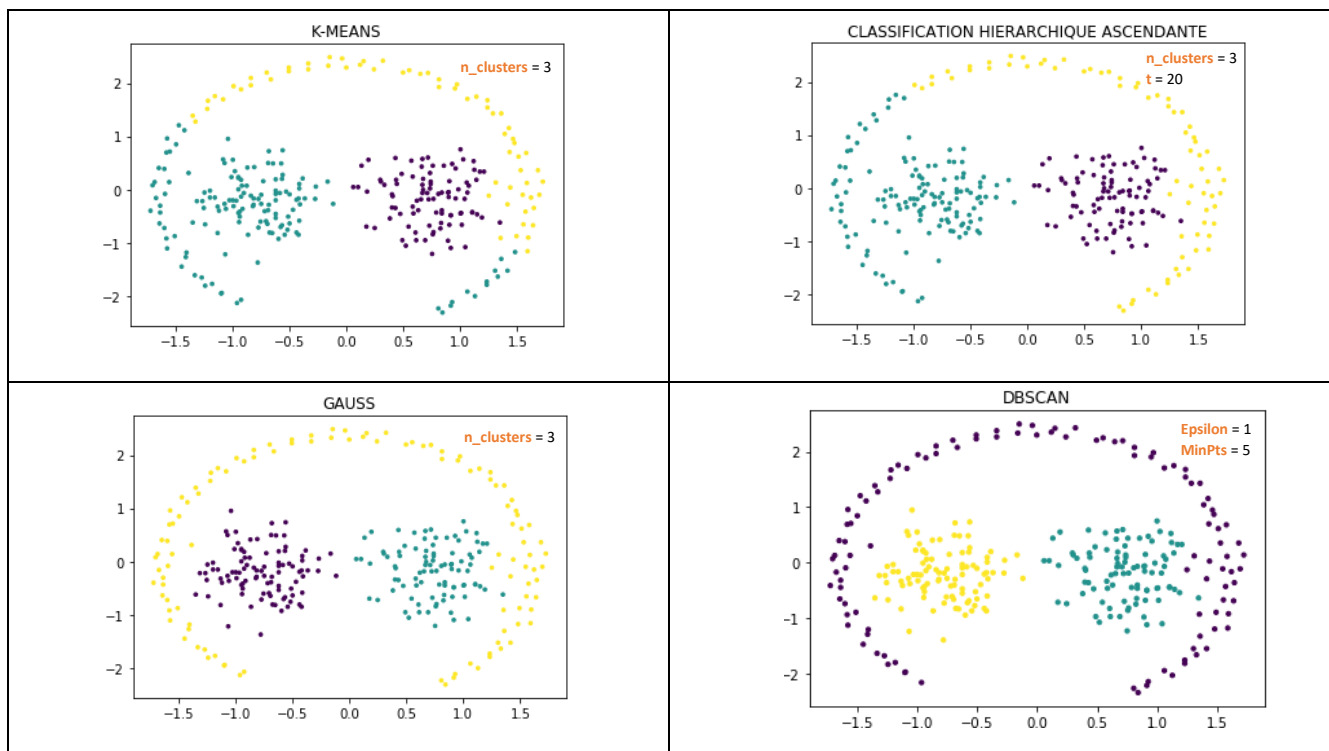
• G20



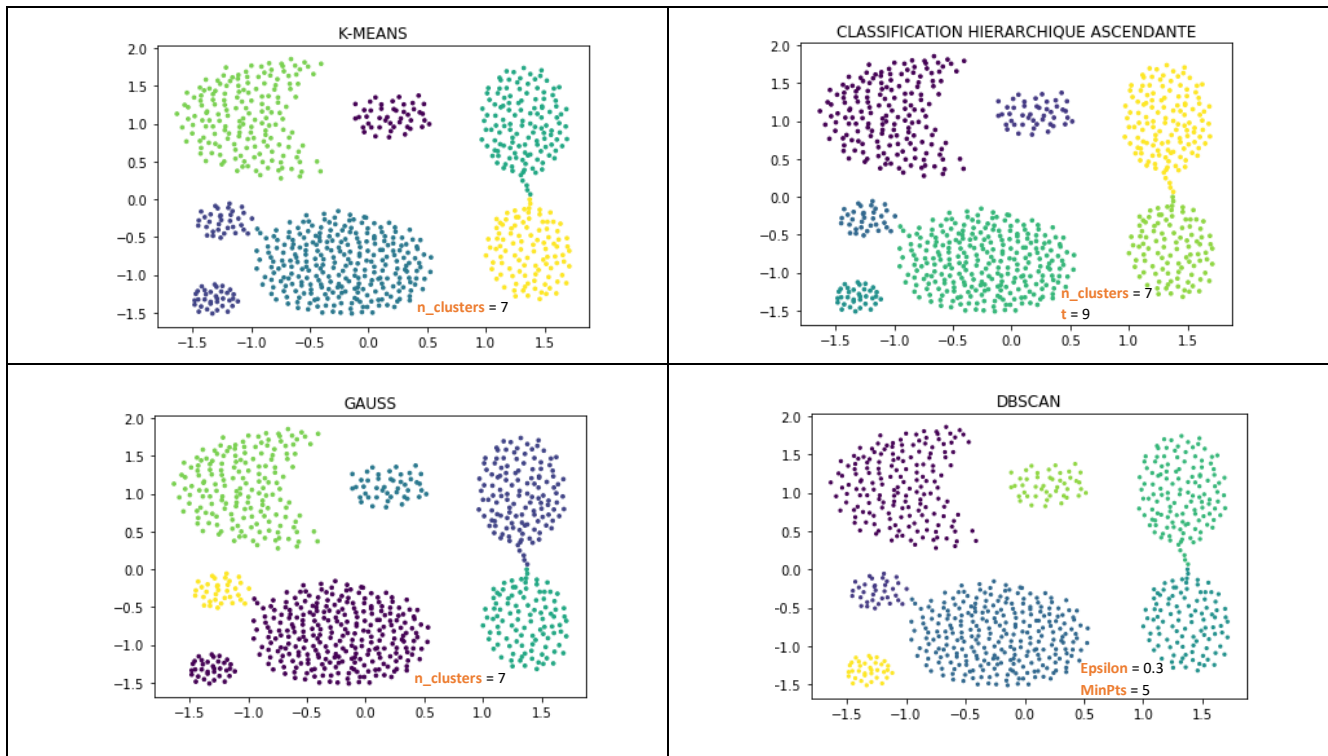
- Jain



- PathBased



- **Aggregation**



ETUDE DES PRECISIONS (par rapport à la vraie partition lorsqu'elle est connue)

Calcul de l'indice de Rand entre la partition trouvée avec l'algorithme et la vérité terrain.

Avec **n_init** = 10 par défaut :

	K-Means	CAH	Gauss	DBSCAN
Pathbased	0.63	0.68	1.0	1.0
Aggregation	0.99	1.0	0.92	1.0
Jain	1.0	1.0	1.0	1.0

Avec **n_init** = 100:

	K-Means	Gauss
Pathbased	0.64	1.0
Aggregation	0.99	0.99
Jain	1.0	1.0

On remarque que certains algorithmes obtiennent de bons résultats. Pour obtenir une meilleure précision, on peut augmenter la valeur du nombre d'exécutions, l'algorithme renvoie alors la meilleure en termes de précision. D'autres en revanche ont du mal à trouver le bon partitionnement sur certains jeux de données (L'algorithme K-means particulièrement, ne produit pas de résultats satisfaisants).

COMPARAISON DES PERFORMANCES

Au vu des résultats, on remarque qu'ils sont dépendants de la texture des données : tous les algorithmes trouvent les bons partitionnements quand les clusters sont ronds et uniformes. En revanche, lorsque les clusters ont des formes atypiques (cf. Jain et son cluster circulaire), certains algorithmes sont plus performants. On étudie plus en détail les résultats obtenus et les performances des algorithmes :

- Le jeu de données **G100** n'offre pas beaucoup de perspectives d'études, à mesure qu'il ne comporte qu'un cluster. On remarque tout de même que seul l'algorithme DBSCAN permet d'éliminer les points trop éloignés du centre de gravité du cluster, en les définissant comme du bruit. Avec les autres méthodes, l'ensemble des données est pris en compte.
- Le jeu de données **G20** est aussi relativement peu parlant, puisque la forme ronde des deux clusters permet d'obtenir des résultats satisfaisants avec tous les algorithmes. Encore une fois, le DBSCAN permet d'éliminer certains points jugés peu homogènes aux autres.
- Le jeu de données **Jain** est bien traité par tous les algorithmes malgré la forme concave très atypique des clusters. A noter que les paramètres de l'algorithme DBSCAN doivent être sélectionnés avec précision, autrement un trop grand nombre de points sont définis comme du bruit.
- Le jeu de données **PathBased** est celui qui obtient le moins de résultats satisfaisants. En effet, le cluster circulaire est le plus difficile à déterminer pour les algorithmes qui ne fonctionnent pas par étude de proche en proche (K-means et CAH). En effet, l'algorithme des K-means cherche à minimiser les distances des points au centre de gravité du cluster. Or ici, le cluster étant circulaire, le barycentre est situé au centre du graphe. Cela rend donc impossible pour l'algorithme de déterminer correctement le partitionnement. Cela s'applique aussi pour la classification ascendante hiérarchique.
- Enfin, le jeu de données **Aggregation** présente des clusters pour la plupart bien convexe et ronds. La difficulté vient de leur proximité entre eux. En effet, certains sont liés par quelques données. On pourrait alors penser que la méthode DBSCAN, qui fonctionne de proche en proche, effectuerait des erreurs de partitionnement. Cependant, ce n'est pas le cas; c'est la méthode de distribution de mélange de gaussiennes qui produit le plus d'erreurs.

Ainsi, l'étude de ces jeux de test permettent de caractériser les faiblesses de certains algorithmes de partitionnement. Pour les clusters ronds, toutes les méthodes sont bonnes et on privilégiera donc les algorithmes les plus rapides et optimisés. En revanche, pour des formes complexes de clusters, on préférera la méthode DBSCAN voire la distribution de mélange de gaussiennes. Le seul inconvénient est qu'il est difficile de déterminer la valeur optimale d'**Epsilon** pour le DBSCAN. En effet, la distance entre des variables de dimension supérieure 3 est difficilement évaluable pour l'esprit humain.

3. Classement des principaux pays du monde en fonction de leur développement

Tableau de données

Indice	country	child_mortality	exports	health	imports	income	inflation	life_expectation	total_fertility	GDP
0	Afghanistan	90.2	10	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.1	76.5	2.89	4460
3	Angola	119	62.3	2.85	42.9	5900	22.4	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200
5	Argentina	14.5	18.9	8.1	16	18700	20.9	75.8	2.37	10300
6	Armenia	18.1	20.8	4.4	45.3	6700	7.77	73.3	1.09	3220
7	Australia	4.8	19.8	8.73	20.9	41400	1.16	82	1.93	1e+06
8	Austria	4.3	51.3	11	47.8	43200	0.873	80.5	1.44	46900
9	Azerbaijan	39.2	54.3	5.88	20.7	16000	13.8	69.1	1.92	5840
10	Bahamas	13.8	35	7.89	43.7	22900	-0.393	73.8	1.86	28000
11	Bahrain	8.6	69.5	4.97	50.9	41100	7.44	76	2.16	20700
12	Bangladesh	49.4	16	3.52	21.8	2440	7.14	0	2.33	758
13	Barbados	14.2	39.5	7.97	48.7	15300	0.321	76.7	1.78	16000
14	Belarus	5.5	51.4	5.61	64.5	16200	15.1	70.4	1.49	6030
15	Belgium	4.5	76.4	10.7	74.7	41100	1.88	80	1.86	44400
16	Belize	18.8	58.2	5.2	57.5	7880	1.14	71.4	2.71	4340
17	Benin	111	23.8	4.1	37.2	1820	0.885	61.8	5.36	758
18	Bhutan	42.7	42.5	5.2	70.7	6420	5.99	72.1	2.38	2180
19	Bolivia	46.6	41.2	4.84	34.3	5410	8.78	71.6	3.2	1980
20	Bosnia and Herzegovina	6.9	29.7	11.1	51.3	9720	1.4	76.8	1.31	4610
21	Botswana	52.5	43.6	8.3	51.3	13300	8.92	57.1	2.88	6350
22	Brazil	19.8	10.7	9.01	11.8	14500	8.41	74.2	1.8	11200
23	Brunei	10.5	67.4	2.84	28	80600	16.7	77.1	1.84	35300
24	Bulgaria	10.8	50.2	6.87	53	15300	1.11	73.9	1.57	6840
25	Burkina Faso	116	19.2	6.74	29.6	1430	6.81	57.9	5.87	575
26	Burundi	93.6	8.92	11.6	39.2	764	12.3	57.7	6.26	231
27	Cambodia	44.4	54.1	5.68	59.5	2520	3.12	66.1	2.88	786
28	Cameroon	108	22.2	5.13	27	2660	1.91	57.3	5.11	1310
29	Canada	5.6	29.1	11.3	31	40700	2.87	81.3	1.63	47400
30	Cape Verde	26.5	32.7	4.09	61.8	5830	0.595	72.5	2.67	3310
31	Central African	140	11.0	2.00	26.5	000	2.01	47.5	5.71	446

Informations sur les données

Data columns (total 10 columns):		
country	167 non-null	object
child_mortality	167 non-null	float64
exports	167 non-null	float64
health	167 non-null	float64
imports	167 non-null	float64
income	167 non-null	int64
inflation	167 non-null	float64
life_expectation	167 non-null	float64
total_fertility	165 non-null	float64
GDP	165 non-null	float64

Partie 3.1 EXAMEN DES DONNEES

Le jeu de données réel est un DataFrame composé de 167 pays décrits selon 9 attributs. Ces critères de description sont choisis de sorte à déterminer globalement "l'état de santé" des pays d'un point de vue socio-économique :

4 caractéristiques de santé publique	5 caractéristiques économiques
<ul style="list-style-type: none"> - Child mortality - Health - Life expectation - Total fertility 	<ul style="list-style-type: none"> - Exports - Imports - Income - Inflation - GDP

Ce DataFrame contient donc 1503 données "utiles" (on ne compte pas les noms des pays, qui vont être enlevés pour le traitement du tableau). De ce fait, c'est un jeu de données relativement petit qui ne va pas poser de problèmes au niveau du temps d'exécution des analyses.

En premier lieu, il convient d'analyser la qualité des données.

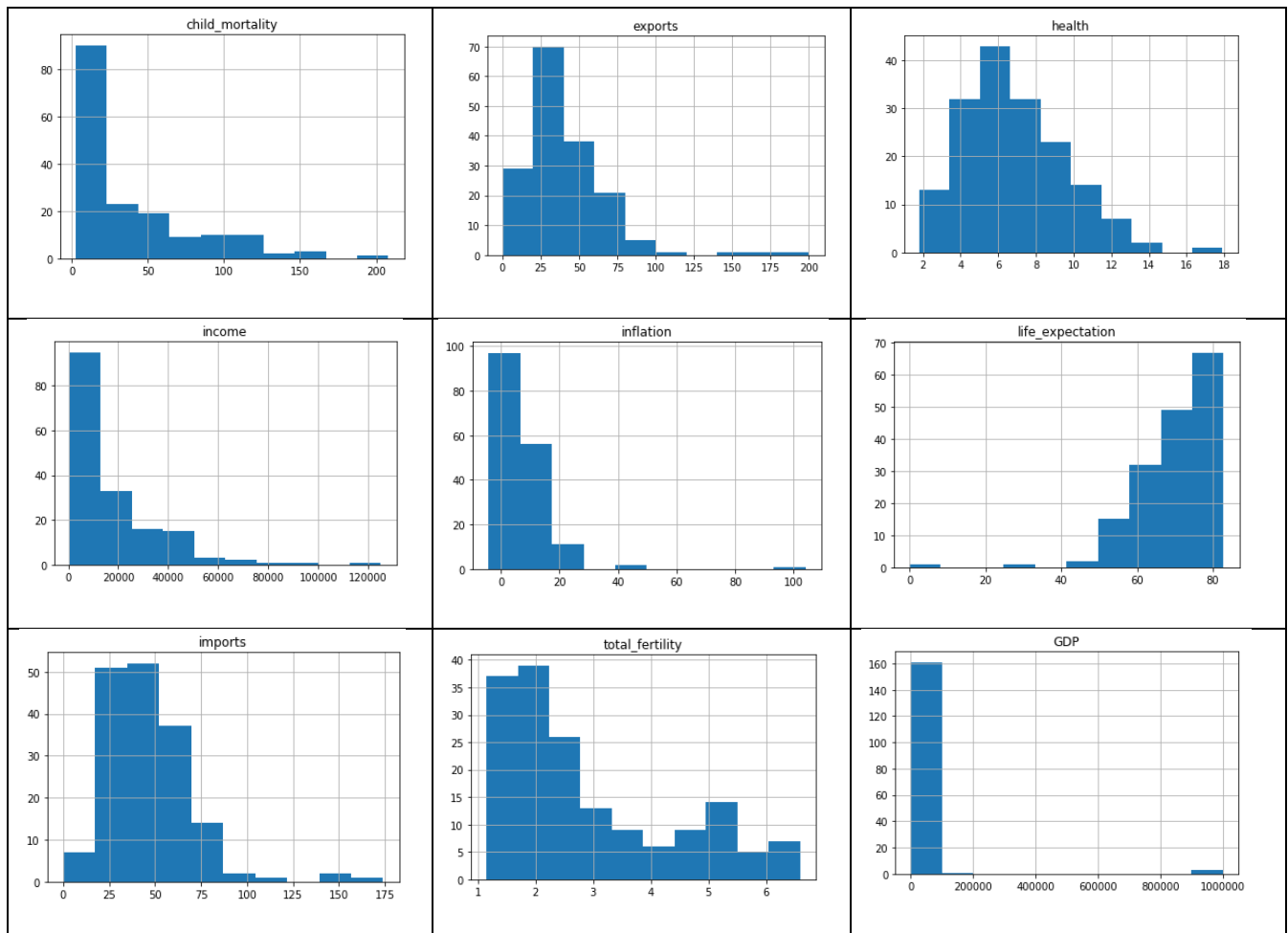
Pour se faire, on utilise d'abord la fonction `describe()` qui renvoie les caractéristiques primaires de chaque colonne (nombre d'individus, moyenne, déviation des données par rapport à la moyenne, minimum, maximum, premier quartile, médiane, troisième quartile).

Indice	child_mortality	exports	health	imports	income	inflation	life_expectation	total_fertility	GDP
count	167	167	167	167	167	167	167	165	165
mean	38.2701	41.109	6.81569	46.8902	17144.7	7.78183	70.1341	2.926	29710.4
std	40.3289	27.412	2.74684	24.2096	19278.1	10.5707	10.4354	1.47928	133495
min	2.6	0.109	1.81	0.0659	609	-4.21	0	1.15	231
25%	8.25	23.8	4.92	30.2	3355	1.81	64.95	1.79	1310
50%	19.3	35	6.32	43.3	9960	5.39	73.1	2.41	4610
75%	62.1	51.35	8.6	58.75	22800	10.75	76.8	3.85	13500
max	208	200	17.9	174	125000	104	82.8	6.59	1e+06

Cette fonction met en lumière des anomalies :

- Il manque 2 données dans les colonnes total_fertility et GDP
- Il y a des valeurs aberrantes (exemples flagrants : un PIB à 1 Million et une espérance de vie de 0)

On se propose alors d'étudier plus en détail les anomalies décelées. Pour se faire, on va afficher les histogrammes de chaque colonne. Cela permet de déterminer plus précisément les valeurs aberrantes. On utilise la fonction `pd.DataFrame.hist()` :



L'étude de ces histogrammes nous permet de confirmer des valeurs aberrantes, qu'il faudra traiter avant de pouvoir étudier le tableau, pour les caractéristiques suivantes :

- Life_expectation
- GDP

Remarque : D'autres valeurs peuvent sembler étonnantes, comme pour les exportations de biens et services. Cependant, de telles valeurs (> 100) sont possibles pour des petits pays qui produisent beaucoup de richesses liées à des facteurs extérieurs. Par exemple, le Luxembourg produit de la richesse via les impôts sur des sociétés étrangères qui s'y installent fiscalement. De même, le Bahreïn exporte du pétrole grâce à des investissements d'entreprises étrangères qui financent une grande partie de l'exploitation. De ce fait, la richesse produite est liée à un apport financier extérieur, ce qui booste artificiellement le pourcentage d'exportations par rapport au PIB.

Partie 3.2 PREPARATION DES DONNEES

TRAITEMENT DES VALEURS MANQUANTES

Le faible nombre de données manquantes (4 au total) nous permet d'effectuer une recherche sur Internet pour un remplacement par des valeurs exactes, qui sont accessibles pour tous sur le site web de la Banque Mondiale (<https://donnees.banquemondiale.org>).

Ainsi, on effectue le remplissage suivant :

Total_fertility	GDP (\$/hab)
France : 1,92 Niger : 7,00	Italy : 34 483 Norway : 81 697

TRAITEMENT DES VALEURS ABERRANTES

Pour modifier les valeurs aberrantes, il est important de comprendre quelles sont les caractéristiques pertinentes à prendre en compte. En effet, il est peu judicieux de remplacer une valeur par la moyenne de la colonne, si celle-ci dépend fortement d'une autre caractéristique (exemple extrême : si une caractéristique B est totalement définie par une autre A, il faut regarder la valeur de cette donnée en A pour déterminer la valeur de la donnée en B). De même, certaines données sont directement accessibles en ligne, ce qui permet de compléter le tableau de manière exacte.

Ici, les valeurs aberrantes sont en petit nombre (5) et sont donc directement remplacées par les valeurs exactes, venant du site de la Banque Mondiale (<https://donnees.banquemondiale.org>) :

Life_expectation	GDP (\$/hab)
Bangladesh : 72	United States : 65 118 United Kingdom : 42 300 Australia : 54 907

Désormais, le tableau est "nettoyé" et présente des données cohérentes avec la réalité. Mais il n'est pas pour autant utilisable tel quel pour les algorithmes de partitionnement. En effet, il faut d'abord enlever la colonne donnant les noms des pays. Ensuite, on centre et réduit l'ensemble des données, via la fonction `StandardScaler()`.

Remarque : le centrage et la réduction sont effectués dans les fonctions implémentées des algorithmes de partitionnement.

Partie 3.3 RECHERCHE DE CORRELATIONS

Index	child_mortality	exports	health	imports	income	inflation	life_expectation	total_fertility	GDP
child_morta...	1	-0.318093	-0.200402	-0.127211	-0.524315	0.288276	-0.886308	0.849227	-0.482742
exports	-0.318093	1	-0.114408	0.737381	0.516784	-0.107294	0.315295	-0.320124	0.408531
health	-0.200402	-0.114408	1	0.0957167	0.129579	-0.255376	0.209375	-0.197349	0.364481
imports	-0.127211	0.737381	0.0957167	1	0.122406	-0.246994	0.0532631	-0.159728	0.107592
income	-0.524315	0.516784	0.129579	0.122406	1	-0.147756	0.611088	-0.503289	0.894327
inflation	0.288276	-0.107294	-0.255376	-0.246994	-0.147756	1	-0.239752	0.319851	-0.223748
life_expect...	-0.886308	0.315295	0.209375	0.0532631	0.611088	-0.239752	1	-0.763234	0.598816
total_ferti...	0.849227	-0.320124	-0.197349	-0.159728	-0.503289	0.319851	-0.763234	1	-0.456104
GDP	-0.482742	0.408531	0.364481	0.107592	0.894327	-0.223748	0.598816	-0.456104	1

De manière générale, les caractéristiques économiques comme les caractéristiques sociales sont corrélées entre elles. Cela est très logique, puisque la santé financière (resp. démographique) d'un pays s'explique par la combinaison des caractéristiques économiques (resp. sociales). A contrario, les couples croisant les caractéristiques économiques et sociales sont peu corrélés, pour les mêmes raisons.

VARIABLES CORRELEES POSITIVEMENT

On retrouve logiquement certains couples de variables très fortement corrélés. Par exemple, il est normal d'avoir une forte corrélation de (total_fertility ; child_mortality) : il est évident (et très malheureux par ailleurs) que moins les enfants ont une chance d'atteindre l'âge adulte, plus le nombre de naissances par couple augmente, et ce afin d'être "assuré" d'avoir une famille.

De même, d'un point de vue économique, il est logique de retrouver le couple (GDP, income) comme étant fortement corrélé : le PIB par habitant étant une mesure de la quantité de richesse produite en moyenne par un habitant, il est normal d'obtenir un salaire élevé lorsque l'on produit beaucoup de richesses.

VARIABLES CORRELEES NEGATIVEMENT

On retrouve également des couples très corrélés négativement, comme (life_expectation ; child_mortality) et (life_expectation ; total_fertility). Cela s'explique simplement par le fait que l'espérance de vie dans un pays traduit les conditions sanitaires de celui-ci (famine, maladies, sécheresses extrêmes, etc.). Les personnes âgées comme les enfants étant des individus fragiles, l'impact de ces conditions les affectent tout particulièrement, ce qui se traduit par une évolution conjointe de ces deux caractéristiques. Et comme on a vu plus tôt que la mortalité infantile est fortement corrélée positivement avec le taux de fécondité, cette caractéristique est par transitivité fortement corrélée avec l'espérance de vie.

Partie 3.4 CLUSTERING DES DONNEES

CHOIX DES PARAMETRES

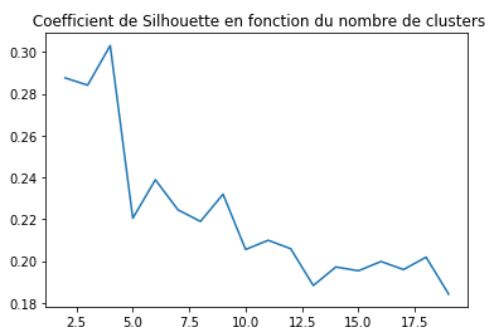
Pour l'ensemble des méthodes, on reprend les mêmes paramètres que pour les jeux de données de la partie 1. Néanmoins, pour un gain de précision, on prend en compte dans certains cas un nombre d'exécutions supérieur à la valeur par défaut.

EVALUATION DE LA QUALITE DES PARTITIONS EN FONCTION DES PARAMETRES

	KMEANS	GAUSS	CAH	DBSCAN
Coefficient de silhouette maximal	0.30	0.20	Etude du Dendrogramme *	0.15
Paramètres	K=4	K=4	K=3 t=18	Eps = 1.2 Min_samples=4 :

- **K-means** (K = 4 / **init** = k-means++ / **n_init** = 100)

Pour cet algorithme, on choisit le nombre de clusters selon la valeur maximale du coefficient de silhouette, c'est-à-dire K = 4. En effet, nous avons choisi celui-ci comme indicateur de la qualité de la représentation.



On choisit également une initialisation selon la méthode k-means++ car cela accélère la convergence de l'algorithme. Ici cependant, le jeu de données est petit et cela n'a que peu d'importance.

Enfin, contrairement aux jeux de données de la partie 1, on fixe le nombre d'exécutions à 100 afin d'obtenir des résultats plus précis, tout en conservant une certaine rapidité de l'algorithme.

- **Mélange des gaussiennes** (K = 4 / **n_init** = 10 / **max_iter** = 1000)

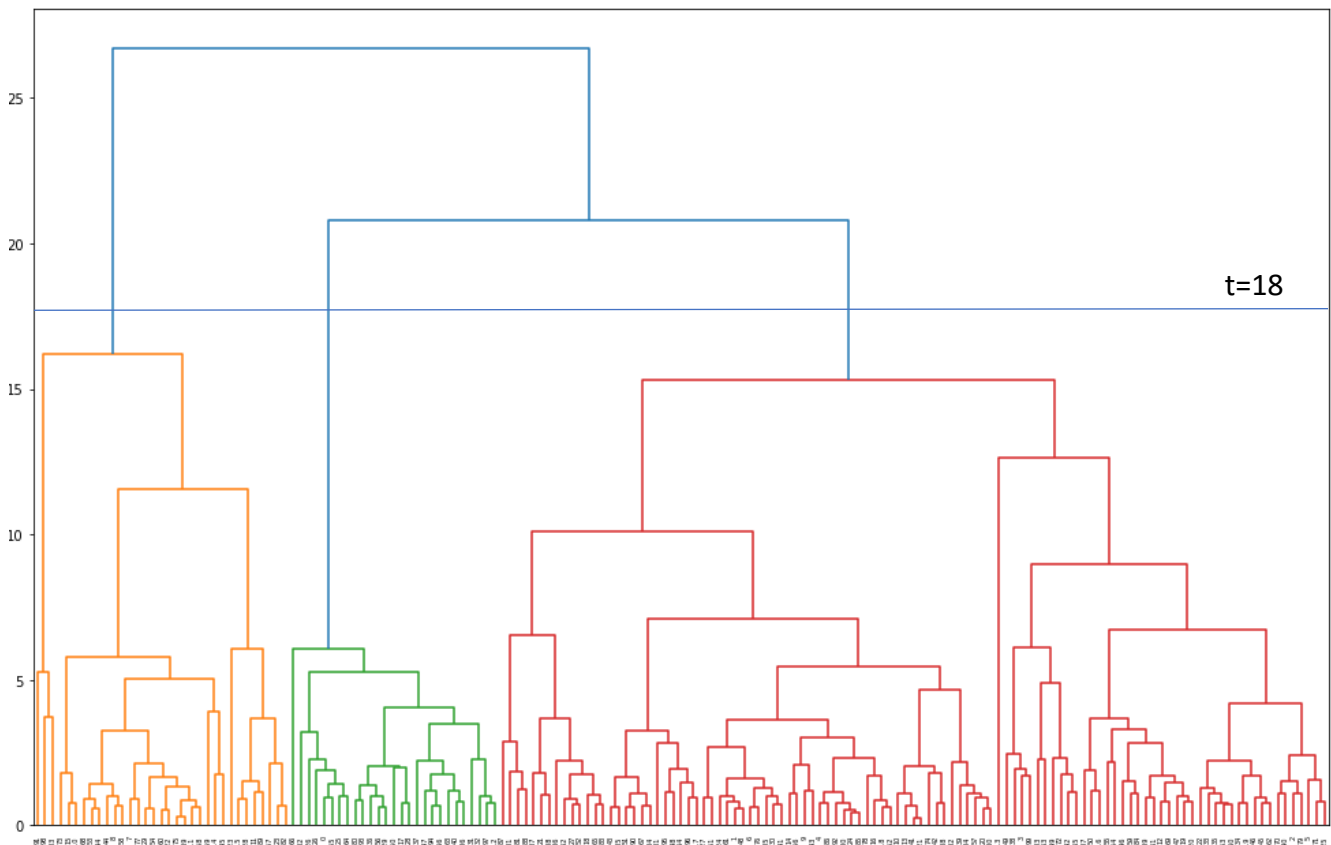
Avec le même raisonnement que pour la méthode des K-means, on détermine à 4 le nombre de clusters.

On choisit un nombre d'initialisations à 10 pour augmenter la précision des résultats en sortie (les meilleurs sont gardés automatiquement).

Enfin, on choisit un maximum de 1000 itérations (10 fois plus que par défaut), afin également d'augmenter la précision des résultats.

- **Classification ascendante hiérarchique** ($t = 18$) *

On choisit le seuil égal à 18 après lecture du dendrogramme associé. En effet, cela permet d'avoir 3 clusters bien différenciés. Le cluster qui importe est celui qui est en vert : il est composé de tous les pays nécessitant prioritairement une aide internationale. De ce fait, il est en réalité possible de prendre n'importe quel seuil supérieur à 6 et inférieur à 20, mais par soucis de clarté et de simplicité, une découpe en 3 clusters est préférable.



- **DBSCAN** ($\text{eps} = 1,2$ / $\text{min_samples} = 4$)

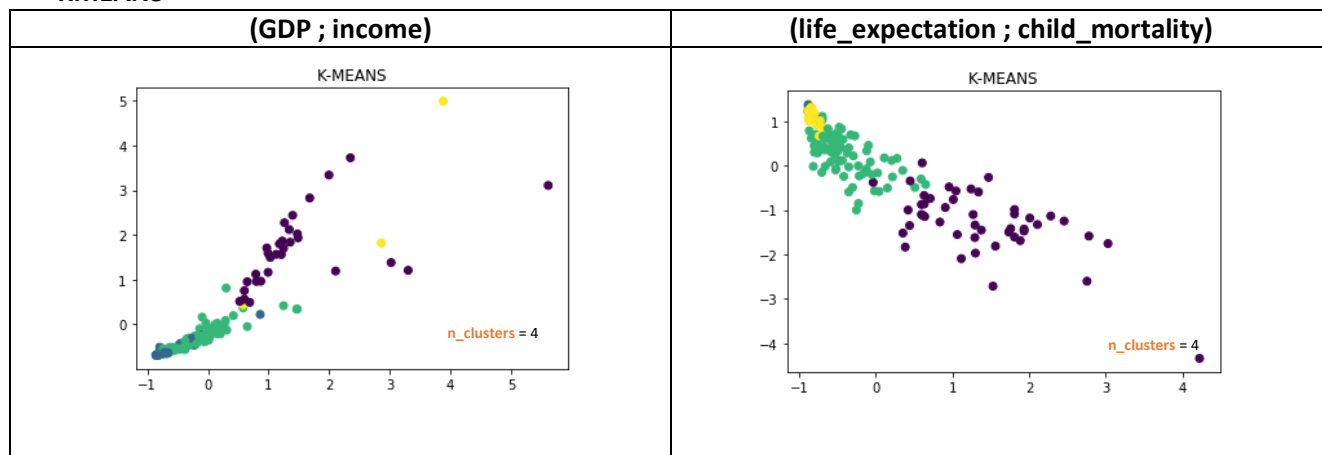
Ici on choisit une valeur d'Epsilon égale à 1,2, car c'est celle qui réalise un bon compromis entre affectation au bruit et partitionnement. En effet, des valeurs supérieures à 1,2 regroupent toutes les données en un seul cluster, et des valeurs inférieures éliminent trop grand nombre de pays.

C'est d'ailleurs pour cette raison que nous n'utiliserons pas cet algorithme pour déterminer la liste des pays dans le besoin : il est illogique d'éliminer de la classification des pays qui ne sont pas homogènes aux autres, car en faisant cela, il est possible que des pays qui sont beaucoup dépendant des aides internationales soient éliminés.

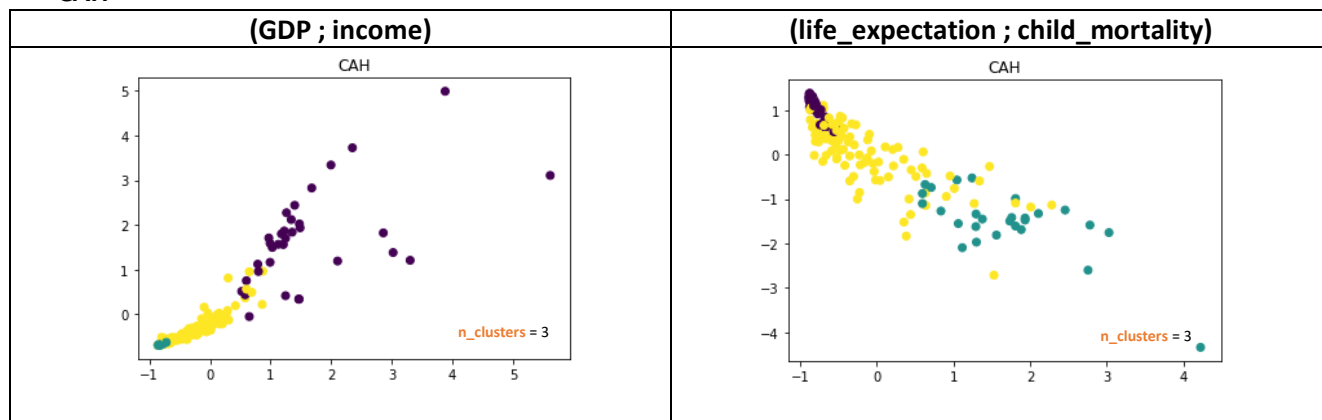
VISUALISATION DES PARTITIONNEMENTS POUR CERTAINS COUPLES DE VARIABLES

Pour visualiser graphiquement les clusters en deux dimensions, on décide de sélectionner les couples de variables les plus corrélées entre elles : (GDP, income) et (life_expectation ; child_mortality). Néanmoins, on s'intéresse davantage à l'étiquetage des données et à leur répartition sous forme de listes de pays plutôt que des évaluations visuelles.

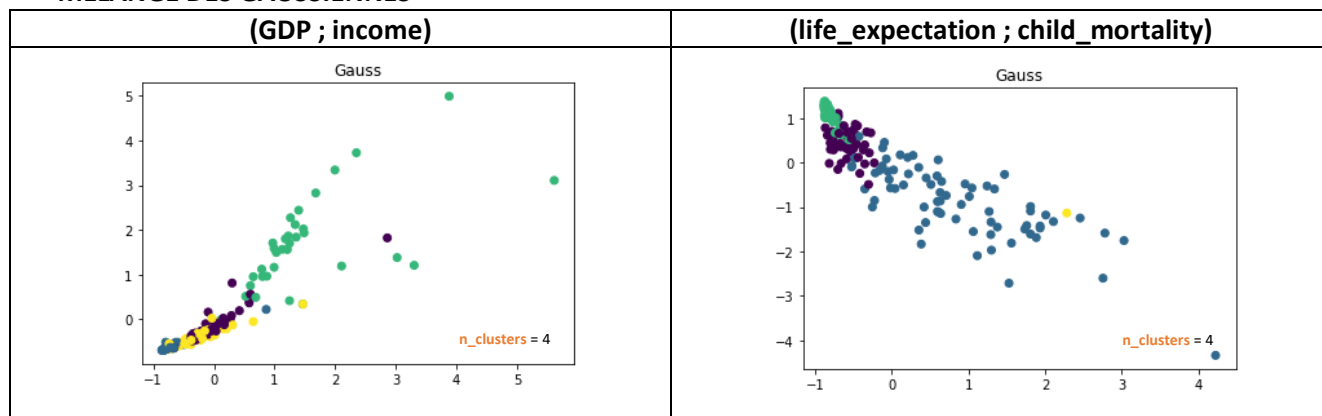
• KMEANS



• CAH



• MELANGE DES GAUSSIENNES

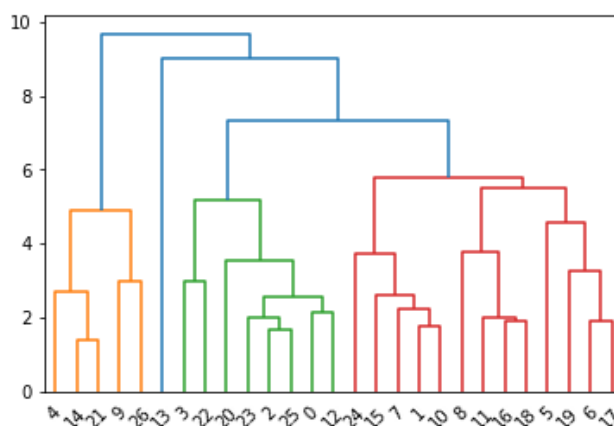


RESULTATS PRELIMINAIRES

En utilisant les 3 algorithmes de tri (tous sauf DBSCAN pour les raisons évoquées plus haut), et en croisant les résultats, on trouve un cluster composé de 27 pays qui sont des candidats potentiels à l'aide internationale. En utilisant les 3 algorithmes de tri (tous sauf DBSCAN pour les raisons évoquées plus haut), et en croisant les résultats, on trouve un cluster composé de 27 pays qui sont des candidats potentiels à l'aide internationale.

On remarque par ailleurs que ces 27 pays sont déterminés intégralement via la méthode par classification hiérarchique (ce cluster de 27 trouvé par CAH est inclus un cluster de chacune des autres méthodes).

De ce fait, on peut déduire que la méthode la plus efficace pour ce jeu de données est la méthode par classification hiérarchique ascendante. On se propose alors d'étudier ce cluster de 27 pays plus en détail avec cet algorithme. Pour se faire, on réalise une classification hiérarchique ascendante sur un DataFrame composé uniquement des 27 pays sélectionnés :



Il ressort en premier lieu que Haïti (numéro 13) est mis à part des autres. En regardant les données, on comprend qu'il est le pays le plus dans le besoin au niveau des aides internationales (plus haute mortalité infantile, plus basse espérance de vie, etc.).

On remarque également que le groupe orange est composé de pays avec un plus fort PIB, et un haut salaire moyen. Ces pays sont donc théoriquement moins prioritaires.

Le groupe rouge, quant à lui, est composé de pays avec un salaire moyen bas. Ce sont également des pays qui effectuent des dépenses très faibles au niveau de la santé, et qui ont une forte mortalité infantile. On peut donc penser que ces pays sont dans le besoin.

Le groupe vert est composé de pays plutôt homogènes, qui ne présentent pas de caractéristiques globales extrêmes.

Ainsi, on s'intéresse en priorité à Haïti et au groupe rouge. Cependant, cela constitue encore trop de pays (14).

En effectuant un découpage plus fin (pour $t = 5,5$), on différencie 3 groupes au sein de ce cluster rouge, dont :

- Un groupe avec une très forte mortalité infantile (groupe de droite)
- Un groupe avec un très faible revenu moyen (groupe du milieu)

De ce fait, on peut dire que ces deux groupes ainsi qu'Haïti sont des pays dans le plus grand besoin.

On a ainsi la liste des 9 pays devant recevoir prioritairement l'aide internationale :

- R.D.C.
- Guinée
- Malawi
- Mozambique
- République Centrafricaine
- Tchad
- Mali
- Niger
- Haïti

Partie 3.5 CLUSTERING DES DONNEES APRES REDUCTION DE DIMENSION

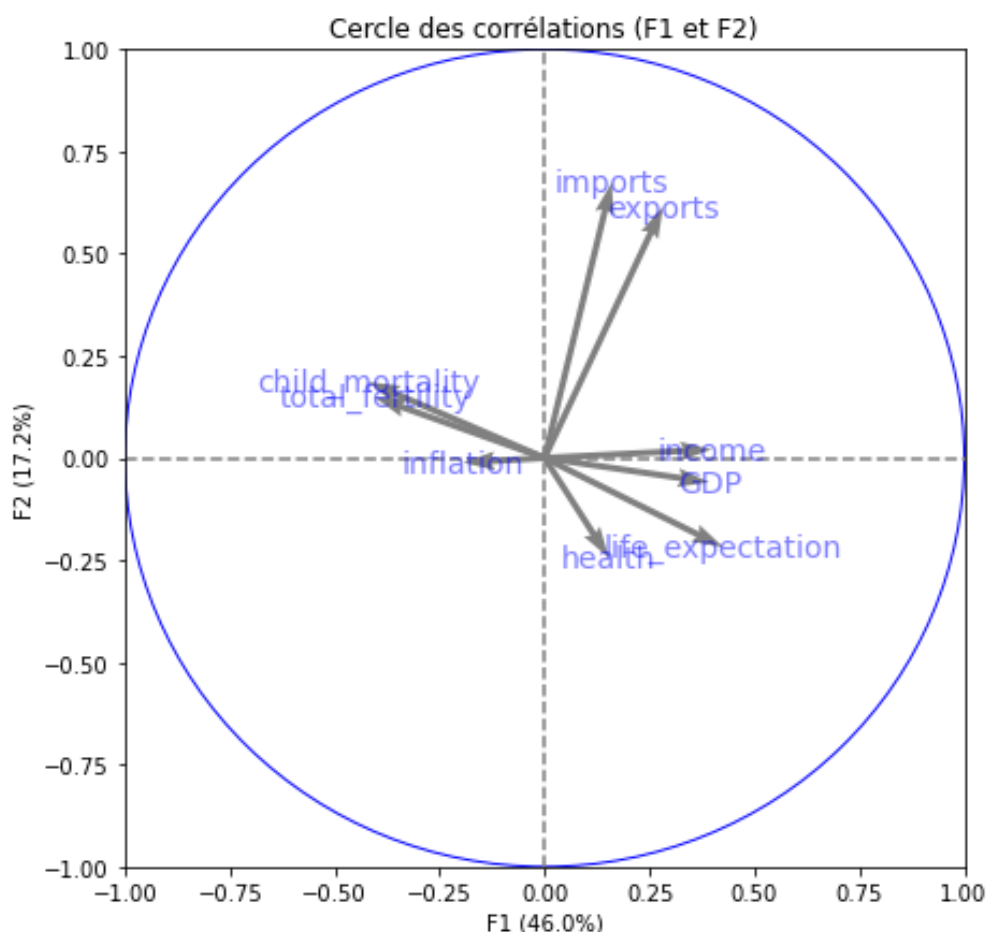
L'étude des variables sur un grand nombre de dimensions reste relativement difficile et peut s'avérer peu précis. De ce fait, on se propose de réaliser une ACP du jeu de données initial et d'observer les éventuels changements au niveau du clustering.

L'ACP est réalisée en gardant 4 composantes principales. Ce nombre correspond à celui qui maximise le pourcentage d'inertie sur le plan principal. Selon le critère de la Part d'Inertie, on garde les axes ayant une part d'inertie supérieure à 0.1 :

Part inertie pour chaque axe : [0.45951011 0.17206082 0.129763 0.11153934 0.07310363 0.02458884 0.01247799 0.00981871 0.00713755]

On réalise alors maintenant l'étude de classification selon les 3 méthodes utilisées précédemment.

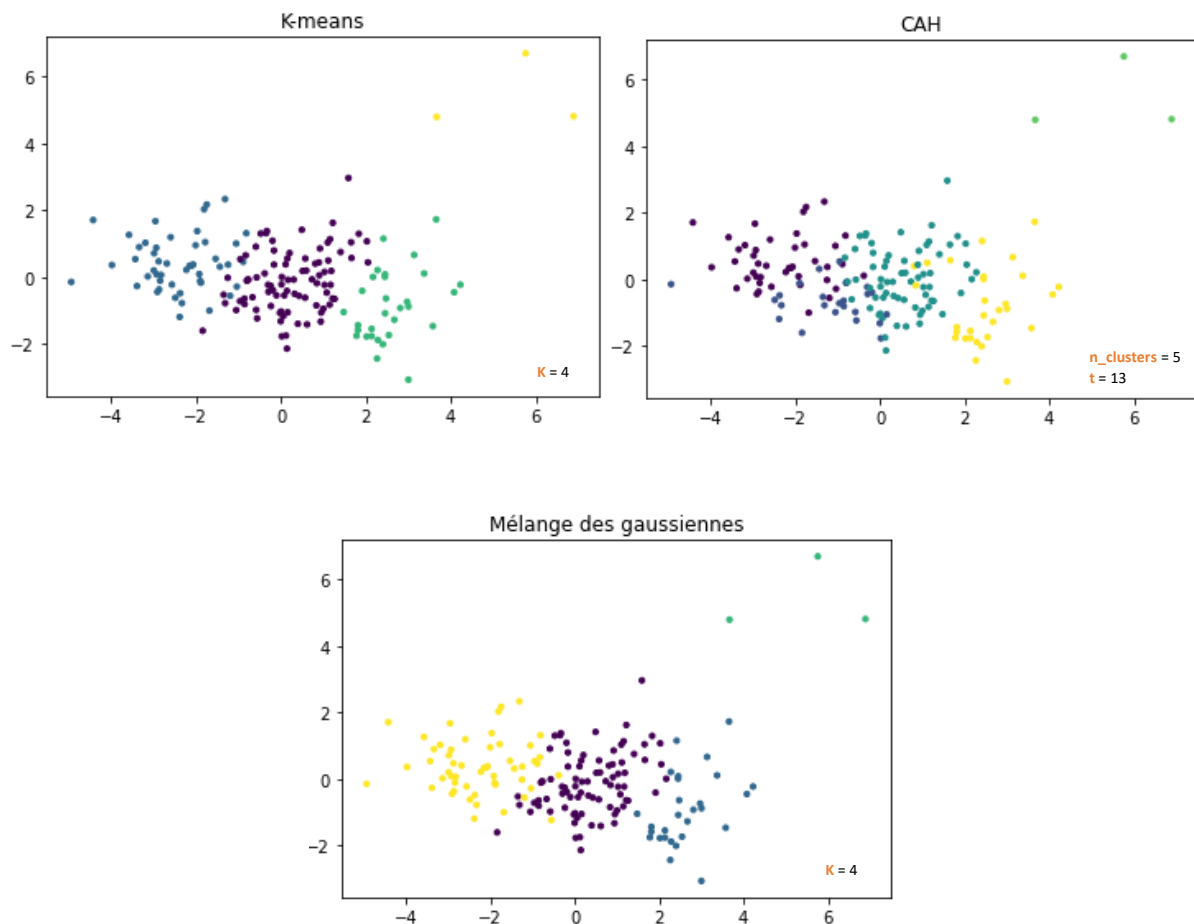
Premièrement, on projette les données dans le plan factoriel. Celui-ci est interprétable via le cercle de corrélation que l'on peut tracer grâce à une fonction `display_circles()` récupérée (et légèrement modifiée) sur le site OpenClassrooms.com.



Ce cercle nous montre, pour le plan factoriel, la contribution des variables aux axes. Ainsi, l'axe vertical est fortement décrit par des variables économiques, tandis que l'axe horizontal est décrit par des variables de mortalité (infantile et espérance de vie), mais aussi par des variables économiques.

PROJECTION DES DONNEES SUR LE PLAN PRINCIPAL

Après avoir appliqué les méthodes de clustering sur les nouvelles données, on projette celles-ci sur le premier plan principal défini par les deux premiers axes principaux :

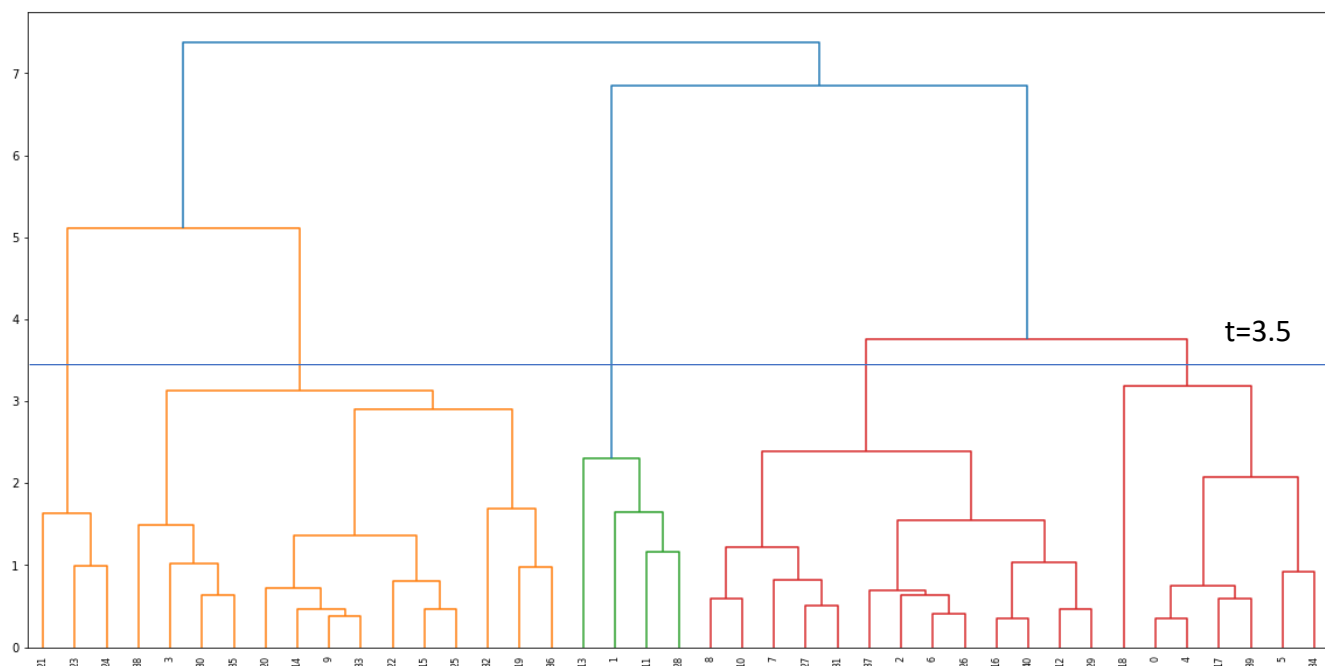


On observe alors des clusters assez homogènes et logiques. Les trois données en haut à droite sont des pays qui importent et exportent beaucoup, tout en ayant un PIB/hab élevé. Ce sont Singapour, le Luxembourg et Malte.

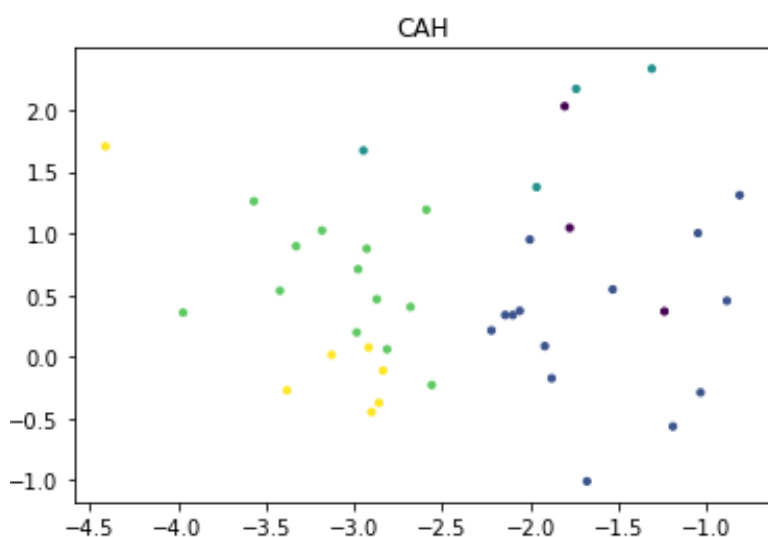
Les pays qui nous intéressent sont ceux situés à gauche du graphe et relativement autour de la hauteur 0. En effet les pays dans le besoin ont une forte mortalité infantile, et des dépenses de santé faibles, sans pour autant avoir beaucoup d'imports/exports. C'est donc le cluster tout à gauche dans chacun des blocs.

Remarque : Il est facile de déterminer à quel pays correspond chaque cluster, puisque la fonction `plt.scatter()` détermine les couleurs par valeur de longueur d'onde : de violet pour le premier cluster, à rouge pour le dernier. Ainsi par exemple, en regardant la liste des clusters, et si les points recherchés sont violets, on prend le premier élément de la liste.

Toutes les méthodes semblent obtenir de bons résultats, et on se propose donc de croiser les données des clusters ciblés. On se retrouve alors avec une liste de 41 pays prioritaires. Pour ceux-ci, on réalise une classification ascendante hiérarchique (via la fonction `cah_reduit()`) et on observe le dendrogramme associé :



Pour avoir un nombre de clusters suffisant, on coupe en $t = 3,5$. Cela nous permet d'obtenir le nuage et le partitionnement suivant :



```
Out[3]:
[['Kiribati', 'Lesotho', 'Liberia'],
 ['Botswana',
  'Comoros',
  'Gambia',
  'Ghana',
  'Iraq',
  'Kenya',
  'Lao',
  'Madagascar',
  'Namibia',
  'Rwanda',
  'Senegal',
  'Solomon Islands',
  'South Africa',
  'Togo'],
 ['Angola', 'Congo Rep.', 'Equatorial Guinea', 'Mauritania'],
 ['Benin',
  'Cameroon',
  'Central African Republic',
  'Chad',
  'Congo Dem. Rep.',
  'Cote d'Ivoire',
  'Guinea',
  'Malawi',
  'Mali',
  'Mozambique',
  'Niger',
  'Tanzania',
  'Zambia'],
 ['Afghanistan',
  'Burkina Faso',
  'Burundi',
  'Guinea-Bissau',
  'Haiti',
  'Sierra Leone',
  'Uganda']]
```

Comme auparavant, ce sont les données à gauche et centrées horizontalement qui sont prioritaires, et donc celles à l'intérieur du cluster vert, avec la donnée jaune tout à gauche, qui correspond à Haïti (sur le dendrogramme, Haïti est la donnée 18 qui est, une fois de plus, mise à part).

Les données vertes sont celles de la partie gauche du groupe rouge sur le dendrogramme (celle qui est entre le numéro 28 et le numéro 18). Au sein de ce groupe, les données les plus à gauche sur le nuage de points sont celles allant de 8 à 31 sur le dendrogramme. Ce sont donc théoriquement ces pays qui nécessiteraient l'aide internationale en priorité, à savoir la liste suivante :

- Tchad
- R.D.C.
- République Centrafricaine
- Mali
- Niger
- Haïti

Conclusion

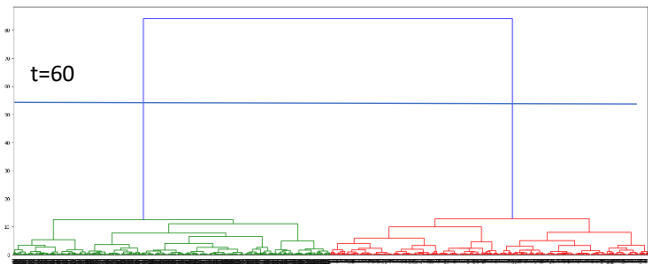
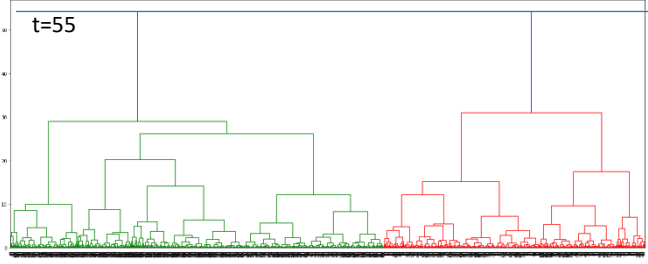
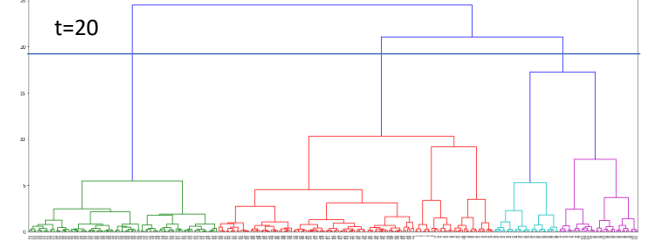
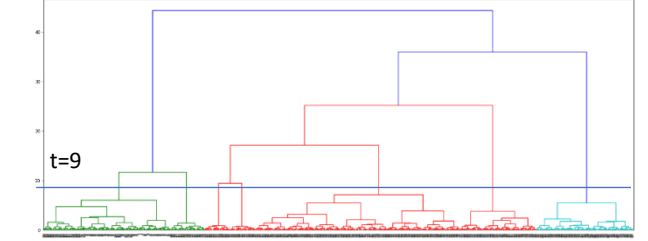
Les deux méthodes utilisées précédemment donnent des résultats extrêmement similaires. En effet, la liste déterminée par l'ACP est incluse dans celle trouvée auparavant. Cela signifie donc que des données ont été éliminées par l'utilisation de cette deuxième méthode. Il est cependant impossible de savoir si ces suppressions sont positives ou négatives, car les pays en question pourraient avoir été noyés dans la 'simplification', ou bien avoir été volontairement écartés.

Néanmoins, il est rassurant de retrouver les mêmes pays dans les deux listes, car cela éloigne grandement la possibilité d'avoir fait des erreurs grossières. De plus, les pays trouvés sont très cohérents avec la situation réelle, car constituent des pays véritablement en manque de ressources essentielles à la survie de leur économie.

ANNEXES

DENDOGRAMMES DES JEUX DE DONNEES TESTS

Nécessaires à l'établissement de la classification hiérarchique ascendante (via la fonction **fcluster**)

<p>G20</p> <p>On garde 2 clusters. On coupe à $t=60$.</p>	
<p>G100</p> <p>On garde 1 cluster. On coupe à $t=55$.</p>	
<p>Pathbased</p> <p>On garde 3 clusters. On coupe à $t=20$.</p>	
<p>Aggregation</p> <p>On garde 7 clusters. On coupe à $t=9$.</p>	
<p>Jain</p> <p>On garde 2 clusters. On coupe à $t=20$.</p>	