

Faculté des sciences

Exploration des méthodes Sparse K-Means et leurs utilisations dans le contexte des données textuelles

Auteur: Jean Marc Ngon

Promoteur: Pr. Johan Segers

Lecteur: Pr. Eugen Pircalabelu

Année académique 2022-2023

Master [120] en science des données, orientation statistique, à finalité spécialisée

Table des matières

Remerciements	iii
Résumé	iv
Abstract	v
Liste des abréviations	ix
1 Introduction générale	1
2 Les défis liés au regroupement des données à hautes dimensions	3
2.1 Les fléaux de la dimensionnalité	3
2.1.1 La malédiction de la dimensionnalité	4
2.2 Réduction de la dimensionnalité comme solution pour lutter contre la dimensionnalité	8
2.2.1 L'analyse en composantes principales	9
2.2.2 t-SNE	11
2.3 Quelques travaux passés sur le regroupement parcimonieux des données . .	13
3 Présentations du K-means et de ses variantes	16
3.1 K-means	16
3.1.1 Algorithme de base	17
3.1.2 La qualité de la partition	18
3.1.3 Choix des paramètres du K-means	19
3.1.4 Quelques inconvénients du K-means	22
3.2 Trimmed K-means	22
3.3 Méthodes de regroupement parcimonieux	22
3.3.1 Motivations	22
3.3.2 Cadre théorique général	24
3.3.3 L'approche Sparse K-means (SK-means)	25
3.3.4 L'approche Robust and Sparse K-mean Clustering (RSKC)	29

4	Applications des algorithmes aux données simulées	31
4.1	Cas A : $n > p$	32
4.1.1	Investigations des méthodes parcimonieuses et comparaisons avec plus de deux variables de regroupement	34
4.1.1.1	Sans valeurs aberrantes	34
4.1.1.2	Avec valeurs aberrantes	37
4.2	Cas B : $n < p$	39
4.2.1	Investigations des méthodes parcimonieuses et comparaisons	40
4.2.1.1	Sans valeurs aberrantes	40
4.2.1.2	Avec valeurs aberrantes	41
5	Application des différentes approches aux données textuelles	44
5.1	Présentation de la base de données	44
5.2	Pré-traitement des données	45
5.3	Représentation vectorielle des données	49
5.3.1	Modèle de Sac des mots (Bag of words)	49
5.3.1.1	La pondération TF-IDF	51
5.4	Applications des différents algorithmes de regroupements	53
5.4.1	Algorithme de K-means	53
5.4.2	Algorithme de Trimmed K-means	55
5.4.3	Algorithme de SK-means	56
5.4.4	Algorithme de RSK-means	57
5.5	Comparaison des valeurs du CER issues des quatre méthodes et sélection de la meilleure méthode	58
5.6	Analyse des classes obtenues	59
6	Discussion/Conclusion	62
A	Code R du chapitre 4	68
B	Code python du chapitre 5	73

Remerciements

Mes remerciement s'adressent aux personnes m'ayant aidé et soutenu de près où de loin dans la réalisation de ce mémoire, ainsi que durant mon parcours à L'UCLOUVAIN.

Je voudrais tout d'abord remercier mon promoteur Pr. Johan Segers, Professeur à l'UCLouvain pour m'avoir accordé sa confiance et offert cette opportunité de réaliser ce travail. Sa disponibilité, ses nombreux conseils et encouragements ainsi que sa relecture et ses remarques ont donné à ce travail une bonne orientation et m'ont booster au moment où je perdais confiance.

Mes remerciement vont également à l'endroit du Professeur Eugen Pircalabelu qui a accepté être lecteur de ce travail.

J'adresse ensuite toute ma gratitude à tout le corps professoral et administratif de l'UCLouvain pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Je remercie ma compagne Mariette pour son soutien moral, financier et matériel tout au long de cette formation.

Je remercie également ma famille et en particulier : ma mère, mes frères, et soeurs pour leurs conseils et soutien moral. Un grand merci à mon cousin Yvan et sa femme pour avoir accepté être mes garants en Belgique.

Enfin je remercie l'ensemble de mes amis du master en sciences des donnée de l'UCLouvain (Armel, Keline, Brice, Olivia, Manuelle etc...) avec qui j'ai passé de bons moments tout au long de mon cursus et qui m'ont apporté des remarques et des conseils lors de la rédaction de ce mémoire.

Résumé

L'émergence des données en haute dimension a suscité un intérêt croissant pour le développement des méthodes d'analyse très efficace tel que le regroupement en classe. Ne connaissant pas toujours la structure de nos données, cette approche est très utile quand l'étiquetage des données devient coûteux. C'est l'exemple d'une large banque d'images ou un grand nombre de documents textuels à classer. Lorsqu'on pense au regroupement en classe, la première méthode qui nous vient en tête est le K-means qui est une approche très simple à mettre en place. Cette étude essaye de combler les limitations du K-means standard en contexte de haute dimension en explorant des méthodes parcimonieuses qui parviennent à identifier les variables cruciales pour la distinction des classes.

Après avoir présenté les premiers travaux de regroupement en classes qui visent à utiliser une technique de réduction de dimension avant le regroupement, nous examinons en détail le K-means traditionnel en mettant en évidence ses avantages et limites dans les espaces de haute dimension. Nous présentons ensuite le trimmed K-means, une version robuste qui fonctionne correctement en présence des valeurs aberrants. Nous présentons également le SK-means et le RSKC, des méthodes parcimonieuses qui ciblent les variables pertinentes pour la distinction des classes.

En appliquant ces méthodes à des données simulées dans diverses configurations, et en utilisant le taux d'erreur de classification (CER) comme métrique d'évaluation, nous comparons leur performance. Les résultats montrent que le RSKC se démarque en tant que méthode la plus performante. Grâce à sa régularisation LASSO et sa robustesse aux valeurs aberrantes, le RSKC identifie efficacement les variables pertinentes responsables du regroupement en classes. Cependant, l'application de ces méthodes à des données textuelles révèle que le trimmed K-means s'adapte étonnamment bien à ce contexte, soulignant son aptitude à discriminer les classes. Ce résultat est expliqué par le fait que le K-means et le Trimmed K-means attribuent des poids égaux aux variables, ce qui peut convenir lorsque les variables ont une importance similaire. Ces résultats mettent en évidence l'importance de choisir la méthode de regroupement en fonction du contexte spécifique des données.

Abstract

The emergence of high-dimensional data has sparked growing interest in the development of highly effective analysis methods, such as clustering. Given that we don't always know the structure of our data, this approach is particularly useful when data labeling becomes costly. This is the case with large image datasets or a substantial number of text documents requiring classification. When considering clustering, the first method that comes to mind is the K-means, which is a straightforward approach to implement. This study aims to address the limitations of standard K-means in a high-dimensional context by exploring sparse methods that manage to identify crucial variables for class differentiation.

After presenting the initial works on cluster analysis that aim to employ dimensionality reduction prior to clustering, we delve into traditional K-means, highlighting its advantages and limitations in high-dimensional spaces. We then introduce the trimmed K-means, a robust version that performs well in the presence of outliers. We also introduce SK-means and RSKC, sparse methods that target relevant variables for class differentiation.

By applying these methods to simulated data in various configurations and using the Classification Error Rate (CER) as an evaluation metric, we compare their performance. The results demonstrate that RSKC stands out as the most effective method. Thanks to its LASSO regularization and robustness to outliers, RSKC efficiently identifies the relevant variables responsible for clustering. However, applying these methods to textual data reveals that trimmed K-means surprisingly adapts well to this context, emphasizing its ability to discriminate between classes. This result is explained by the fact that K-means and trimmed K-means assign equal weights to variables, which can be suitable when variables carry similar importance. These findings underscore the importance of selecting a clustering method based on the specific context of the data.

Table des figures

2.1.1 Représentation de 20 points en 1D,2D et 3D inspiré de [1]	6
2.1.2 Densité du nombre de points par unité de volume	7
2.1.3 Évolution de la taille d'échantillon nécessaire pour garder la même densité de points en fonction du nombre de variables	7
2.1.4 Évaluation de l'écart entre la distance maximale et minimale des points en fonction du nombre de dimensions.	8
2.2.1 Projection des données en utilisant les deux méthodes de réduction de dimension (inspiré du code python de Sciki-learn)	13
3.3.1 Échantillon de données avec 4 classes (source [2])	23
3.3.2 Échantillon de données projetées sur une dimension avec histogramme(tiré de [2])	24
3.3.3 Échantillon de données projetées sur deux dimensions (source [2])	24
4.1.1 Exploration de quelques variables discriminantes dans les données initiales.	33
4.1.2 Nuages de points permettant de comparer graphiquement les performances des différents algorithmes (sans valeurs aberrantes). La partition présente dans les données originales sert de référence.	33
4.1.3 Nuages de points permettant de comparer graphiquement les performances des différents algorithmes (avec valeurs aberrantes). La partition présente dans les données originales sert de référence.	34
4.1.4 Distribution des poids pour la valeur optimale de s	35
4.1.5 Choix des paramètres du sparse k-mmean	36
4.1.6 Choix des paramètres du sparse k-mmean du modèle 1	39
4.1.7 Choix des paramètres du sparse k-mmean du modèle 2	39
4.2.1 Choix des paramètres du sparse k-mmean ($n < p$) sans valeurs aberrantes .	40
4.2.2 Choix des paramètres du Sk-mmean du modèle 1	42
4.2.3 Choix des paramètres du Sk-mmean du modèle 2	42
5.1.1 Présentation des 5 premières lignes de la base de données.	45
5.1.2 Visualisation du nombre de documents par catégorie.	46
5.1.3 Distribution de la longueur des documents : le logarithme du nombre de caractères de chaque documents ou article.	47

5.2.1 Présentation des deux premiers paragraphes du premier document avant nettoyage.	47
5.2.2 Présentation des deux premiers paragraphes du premier document après nettoyage.	49
5.3.1 Présentation des 5 premières lignes obtenues après pondération TF-DF. Rappelons que la matrice entière a une dimension de (1735,5590)	53
5.4.1 t-sne sans labels.	54
5.4.2 T-sne avec les labels obtenus à partir du K-means.	55
5.4.3 t-sne avec les labels obtenus à partir du TK-means.. . . .	56
5.4.4 Distribution des poids pour la valeur optimale de s dans le cas des données textuelles	57
5.4.5 T-sne avec les labels obtenus à partir du SK-means.	58
5.4.6 T-sne avec les labels obtenus à partir du RSK-means.	59
5.6.1 Représentation des mots communs les plus fréquent de chaque classe. . . .	60
5.6.2 Nuage de mots les plus fréquents de chaque classe.	61

Liste des tableaux

4.1.1 Table du nombre de variables non nuls obtenu en combinant les valeurs de s et α	36
4.1.2 Table des valeurs du CER obtenues en combinant les valeurs de s et α . . .	37
4.1.3 Table de comparaison des valeurs du CER et du nombre de variables pertinentes par méthode de regroupement. les valeurs dans les parenthèses représentent les écarts types	37
4.1.4 Table de comparaison des valeurs du CER et du nombre de variables pertinentes par méthode de regroupement. les valeurs dans les parenthèses représentent les écarts types	40
4.2.1 Table de comparaison des valeurs du CER moyen et du nombre moyen de variables pertinentes par méthode de regroupement($n < p$ sans valeurs aberrantes). les valeurs dans les parenthèses représentent les écarts types. Résultats obtenus après 100 simulations de nos données	41
4.2.2 Table de comparaison des valeurs du CER moyen et du nombre moyen de variables pertinentes par méthode de regroupement($n < p$ avec valeurs aberrantes). les valeurs dans les parenthèses représentent les écarts types. Résultats obtenus après 100 simulations de nos données	43
5.3.1 Représentation matricielle du sac des mots. 7 mots du vocabulaire ont été représentés.	51
5.3.2 Représentation matricielle du TF. 7 mots du vocabulaire ont été représentés.	52
5.3.3 Représentation matricielle du IDF. 7 mots du vocabulaire ont été représentés.	52
5.3.4 Représentation matricielle du TF-IDF. 7 mots du vocabulaire ont été représentés.	53
5.4.1 TK-means : CER en fonction de la proportion α	55
5.5.1 Comparaison des différentes méthodes en fonction du CER.	59

Liste des abréviations et définitions

Symbole	Représente.
CER	Classification Error Rate .
SK-means	Sparse K-means.
RSKC	Robust and Sparse K-means Clustering.
LASSO	Least Absolute Shrinkage and Selection Operator.
NLP	Natural Language Processing.
$\mathbf{X}$	Matrice des p caractéristiques pour tous les points de données, appelée matrice de conception à n lignes et p colonnes (x_i est la i ème ligne de $\mathbf{X}$).
$\mathbf{x}$	$\mathbf{x}$ en minuscule et en gras pour désigner un vecteur.
x	x en minuscule pour désigner un scalaire.
$\mathbf{X}^t$	transposé de la matrice $\mathbf{X}$.
PCA	Principal Component Analysis.
t-SNE	t-Distributed Stochastic Neighbor Embedding.
BIC	Bayesian Information Criterion
$\mathbf{X} = (X_1, \dots, X_p)$	vecteur aléatoire de dimension P .
AIC	Akaike Information Criterion.
TSS	Total Sum of Squares.

Chapitre 1

Introduction générale

L'avancement rapide des outils informatiques, ainsi que des progrès de la technologie Internet ont entraîné une collecte et conservation massive des données. Selon les chercheurs, l'expansion moyenne des données double environ tous les 20 mois [3]. Cependant, ces données, souvent brutes au départ et en raison de leur énorme taille, ont des structures cachées qui les rendent presque inutilisables. Elles nécessitent des techniques adéquates afin d'être exploitables en vue d'une meilleure prise de décision. Les chercheurs ont développé un certain nombre d'outils et techniques dans le domaine de l'exploration de données pour déceler les modèles cachés dans ces données. Parmi ces outils, on a l'apprentissage automatique¹ (machine learning) qui a connu une avancée remarquable ces dernières années. Il englobe les problèmes tels que la classification, la régression, regroupement en classe etc. L'analyse de regroupement des données en classes, également connue sous le nom de "clustering" en anglais, est une technique qui vise à identifier des structures sous-jacentes au sein d'un ensemble de données, en regroupant des observations similaires en classes ou en groupes distincts. Cette technique a trouvé de nombreuses applications dans des domaines variés tels que la biologie, la finance, la reconnaissance de formes, et bien d'autres. Cependant dans certaines situations surtout en très haute dimension comme analyse des données textuelle (NLP : Natural Language Processing), il est fort de constater que un bon nombre de dimensions ne participent pas à la discrimination des classes. Il est donc important d'identifier celles qui sont pertinentes dans le processus de regroupement et donc à développer des approches de regroupement parcimonieux. C'est ce challenge majeur qui a poussé un bon nombre d'auteurs à se pencher sur la question.

Dans ce contexte, On est en droit de se poser la question suivante : Les méthodes de regroupement parcimonieux tels que les approches Sparse K-means sont elles efficaces dans le contexte des données creuses à haute dimension ? Ce qui nous amène dans cette étude à faire une analyse et une comparaison de 4 approches de regroupement : le K-means de base, le Trimmed K-means, le SK-means et le RSKC. L'objectif étant de déterminer l'approche la plus adaptée pour identifier des classes significatives ainsi que les caractéristiques discriminantes

1. L'apprentissage automatique est une branche de l'intelligence artificielle (IA) qui se concentre sur le développement de techniques permettant aux ordinateurs d'apprendre à partir d'une grande masse de données.

dans une base de données.

Notre travail s'articule autour de 5 grands axes. Dans le **chapitre 2**, nous présentons quelques défis liés aux regroupement des données à haute dimension. Il s'agit principalement de présenter les problèmes qu'on rencontre lorsqu'on est en haute dimension, de discuter de quelques solutions permettant de remédier à ces challenges. Nous présentons également dans ce chapitre les travaux passés menés par des chercheurs dans le contexte du regroupement en haute dimension. Dans le **chapitre 3**, nous présentons les différents algorithmes à savoir le K-means de base, le Trimmed K-means, le SK-means et le RSKC. Cela nous permet de poser un cadre théorique sur lequel nous nous appuyons afin de mener nos expérimentations. Le **chapitre 4** présente en détail nos expérimentations sur des ensembles de données simulées. Nous examinons comment chaque approche fonctionne dans différentes configurations, en tenant compte des paramètres de chaque méthode. Les résultats obtenus nous permettent de comparer la performance de ces approches en fonction de différentes mesures d'évaluation, tout en soulignant les avantages et les inconvénients spécifiques à chaque méthode. Dans le **chapitre 5**, nous abordons l'application de ces approches aux données textuelles. Nous expliquons en détail le prétraitement des données qui est une étape cruciale dans ce contexte. Ensuite, nous décrivons comment les approches de regroupement sont appliquées à ces données textuelles, en analysant les résultats obtenus et en identifiant les tendances de regroupement dans les différentes approches. Le **dernier chapitre** est dédié à la discussion et à la conclusion.

Ce mémoire vise à fournir une vue d'ensemble des différentes approches de regroupement abordées, en mettant en évidence leurs performances tant dans les données simulées que dans les données textuelles. En explorant à la fois les avantages et les inconvénients de chaque méthode, nous espérons contribuer à la compréhension des choix de méthodes de regroupement et guider les décisions dans la pratique.

Chapitre 2

Les défis liés au regroupement des données à hautes dimensions

Le regroupement en classes est une méthode courante de l'apprentissage non supervisé permettant de regrouper des données similaires en différents groupes ou classes. Son but est de découvrir à la fois les régions denses et les régions clairsemées dans l'ensemble de données. Cependant, à haute dimension, le regroupement des données peut être confronté à des défis particuliers en raison de la nature des données, notamment les données creuses (sparse) dont il est difficile de détecter la structure. Les données creuses à haute dimension sont caractérisées par un grand nombre de dimensions avec un petit nombre de valeurs non nulles ou peu significatives pour effectuer des calculs, ce qui peut avoir un impact significatif dans les résultats. Parmi ces types de données, on peut citer les données issues des systèmes de recommandation basées sur les scores fournis par les utilisateurs¹ ou encore les données issues du traitement automatique du langage naturel³. Dans cette optique, cette partie se concentrera sur les défis spécifiques liés au regroupement des données à haute dimension, notamment le fléau de la dimensionnalité. Les méthodes de réductions de dimension et quelques travaux passés sur les challenges du regroupement en classes seront également abordés.

2.1 Les fléaux de la dimensionnalité

La dimensionnalité d'une base de données fait référence au nombre de caractéristiques ou de variables utilisées pour décrire chaque observation, individu ou objet. Par exemple, dans un espace bidimensionnel, chaque observation est représentée par deux coordonnées (x, y) tandis que dans un espace tridimensionnel, chaque observation est représentée par trois coordonnées (x, y, z) . Lorsque le nombre de caractéristiques devient très grand, on se retrouve dans un cas de haute dimension. Dans l'analyse des données, on distingue

1. Les recommandations des films par exemple se basent sur les scores des films. Il peut arriver que beaucoup de films ne soient pas notés par les utilisateurs ; cela impliquera une forte présence de zéro d'où l'obtention des données creuses

généralement deux situations : le cas où le nombre de caractéristiques est inférieur au nombre d'observations et le cas où le nombre d'observations est inférieur au nombre de caractéristiques. Maintenant, quels sont les effets réels de l'augmentation du nombre de dimensions ? Comment cela influe-t-il sur la détection d'une structure dans le jeu de données ? Nous répondrons à ces deux questions dans les sous-sections suivantes.

2.1.1 La malédiction de la dimensionnalité

Richard Bellman fut celui qui a inventé l'expression 'Curse of Dimensionality' (la malédiction de la dimensionnalité) en 1957 dans son livre intitulé « Dynamic Programming » [4]. Ce concept est employé pour décrire la croissance exceptionnelle de la complexité des algorithmes lorsque le nombre de variables augmente dans les problèmes d'optimisation. Concrètement, cela signifie que lorsque le nombre de variables ou de caractéristiques augmente dans un jeu de données, les mesures de distance deviennent moins significatives et les points s'étendent jusqu'à devenir équidistants [2].

Pour illustrer cette situation, commençons par un exemple simple. Dans la figure 2.1.1, nous avons 20 points répartis le long d'une dimension divisée en 10 cellules (a), ce qui donne un ratio de 2 points par cellule. Lorsque nous ajoutons une dimension (b), les points se dispersent davantage, créant maintenant 10^2 cellules, soit un ratio de 0.2 points par cellule. En ajoutant une troisième dimension, les points se dispersent encore plus, aboutissant à 10^3 cellules avec un ratio de 0.02 points par cellule. On peut clairement constater que l'augmentation du nombre de dimensions, tout en conservant la même taille d'échantillon, entraîne une dispersion plus importante.

La figure 2.1.2 illustre clairement la décroissance rapide du nombre de points par unité de volume. Dans ce cas, nous avons simulé $N = 100$ points répartis selon $\frac{N}{2^n}$, avec $n = 3, \dots, 40$. On peut observer que la densité de points devient presque nulle à partir de 10 dimensions. Puisque les algorithmes d'apprentissage automatique requièrent un nombre considérable d'observations pour fournir de bons résultats, il est crucial d'avoir un équilibre entre le nombre d'observations et les variables. Cependant, il n'existe malheureusement pas de solution simple à ce problème.

L'article [5] propose une piste pour aborder ce problème. Selon les auteurs, on peut approximativement considérer qu'un bon ratio est d'avoir $n - 1$ caractéristiques pour n observations, mais uniquement si les variables ne sont pas corrélées. Si les variables sont fortement corrélées, elles apportent de l'information inutile et redondante. Dans le cas de fortes corrélations, les auteurs suggèrent que le nombre de variables soit égal à la racine carrée du nombre d'observations. Par exemple, cela donnerait un ratio de 30 variables pour 1000 observations.

En réalité il est difficile de se donner de tels standards. Que se passerait-il si on ne les respectait pas ? De combien d'observations avons-nous besoin en fonction des dimensions ? Supposons un cas simple tiré de [6] où nous avons des combinaisons de variable binaires : Nous voulons 10 observations par combinaison unique c'est-à-dire 10 pour chaque valeur 0 et 1. Afin de garder la même densité dans notre espace de données, on a :

— variable binaire $\rightarrow$ 2 combinaisons uniques $\rightarrow$ 20 observations nécessaires.

- 2 variables binaires $\rightarrow$ 4 combinaisons uniques $\rightarrow$ 40 observations nécessaires.
- k variables binaires $\rightarrow 2^k$ combinaisons uniques $\rightarrow 10 \times 2^k$ observations nécessaires.

La figure 2.1.3 montre qu'autour de 17 variables, nous avons besoin de 1 millions d'observations si nous voulons maintenir la même densité de points pour nos analyses ; Un passage de 17 à 20 variables (juste une augmentation de 3 variables) entraîne un passage de 1 million à 10 millions d'observations.

Donc une augmentation linéaire de variables enrènerait une augmentation exponentielle d'observations pour une bonne performance des algorithmes d'apprentissage automatique.

En pratique, une augmentation de la taille de l'échantillon peut s'avérer très coûteuse : Souvent on est limité en ressources : Les plans d'expérience en laboratoire, Interroger plus de personnes lors d'une enquête, prendre plus de mesure, etc ; et on se retrouve souvent face à un très grand nombre de variables ; c'est le cas par exemple de l'analyse des documents textuels où avec seulement 10 documents, on peut se retrouver avec plus de cent milles mots (c'est-à-dire cent milles dimensions). L'auteur de l'article [7] dans son étude conclut qu'en gardant la taille d'échantillon d'entraînement fixe, la performance de tout modèle de classification augmente dans le même ordre que les dimensions, mais à partir d'un certain nombre de dimension, cette performance se détériore. Ce phénomène est connu sous le nom de **Phénomène de Hughes**. Il faut donc trouver des méthodes appropriées pour réduire les dimensions (voir la section suivante).

- L'effet de la dimensionnalité sur la distance entre les points :

Lorsque le nombre de dimensions devient élevé, la mesure des distances devient peu significative. Les points tendent à se répartir de manière équidistante, rendant difficile la détection de tendances ou de structures dans les données. Pour illustrer cette situation, Nous générons 1000 points uniformément distribués dans un espace de dimension variant de 2 à 50. Nous choisissons un point P au hasard comme point de référence et nous calculons sa distance aux autres points de l'espace. Nous calculons ensuite la différence entre la distance maximale (point le plus éloigné de P) et la distance minimale (point le plus proche de P). Ainsi donc par soucis d'échelle on évalue pour chaque nombre de dimensions de 2 à 50, l'expression suivante :

$$(\text{Max}_{\text{dist}} - (\text{Min}_{\text{dist}})) / (\text{Max}_{\text{dist}}).$$

Cette simulation peut être généralisée pour chaque point choisi au hasard dans l'espace. La figure 2.1.4 illustre une décroissance rapide de cette différence, tendant vers zéro. Cela signifie qu'à partir d'un certain nombre de dimensions, $(\text{Max}_{\text{dist}} \approx (\text{Min}_{\text{dist}}))$. Cette observation confirme que les points tendent à se répartir de manière de plus en plus équidistante à mesure que le nombre de dimensions augmente. En conséquence, il devient difficile de distinguer les groupes d'observations sous-jacents.

Un autre point important à souligner est **l'éloignement des points du centre de l'espace quand les dimensions augmentent** ; par conséquent, le volume de l'hyperboule unitaire tend vers zéro [8]. En effet on a :

$$V_r(d) = \frac{2r^d \pi^{d/2}}{\Gamma(1 + d/2)}$$

(Γ est la fonction gamma) le volume de l'hyperboule unitaire dans l'espace de dimension d et de rayon $r = 1$ contenu dans un hypercube unitaire de volume $V_l(d) = l^d = 1$ où l est l'arrête et vaut 1.

Rappelons que $\Gamma(z + 1) = z\Gamma(z)$, $\forall z \geq 0$; le volume de l'hyper-boule devient :

$$V_r(d) = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$$

donc

$$\lim_{d \rightarrow \infty} \frac{\text{Volume de l'hyperboule}}{\text{volume de l'hypercube}} = 0$$

Ceci s'interprète comme s'il n'y avait pas de volume dans l'hyperboule et que tout le volume se concentrait dans les coins de l'hyper-cube. En d'autres termes, la probabilité de trouver un point à l'intérieur de l'hyperboule est presque nulle; tous les points se situent dans les coins de l'hypercube. Ceci constitue un obstacle pour les algorithmes qui dépendent de la localité spatiale des données comme le regroupement en classes [9].

Les conséquences des points su-énumérés sont : **l'existence des variables non pertinentes et des variables fortement corrélées.**

Les variables non pertinentes sont les moins informatives et constituent du bruit. Elles présentent une variation faible causée par l'équidistance des points. Elles ne permettent pas de distinguer les points de manière significative.

En ce qui concerne les variables fortement corrélées, elles ajoutent une information redondante. La présence d'un attribut devient superflue car un autre attribut peut déjà capturer les motifs similaires. Pour plus de détails, voir [9]. Pour palier à ces problèmes, l'une des solutions est de réduire considérablement la dimensionnalité des données.

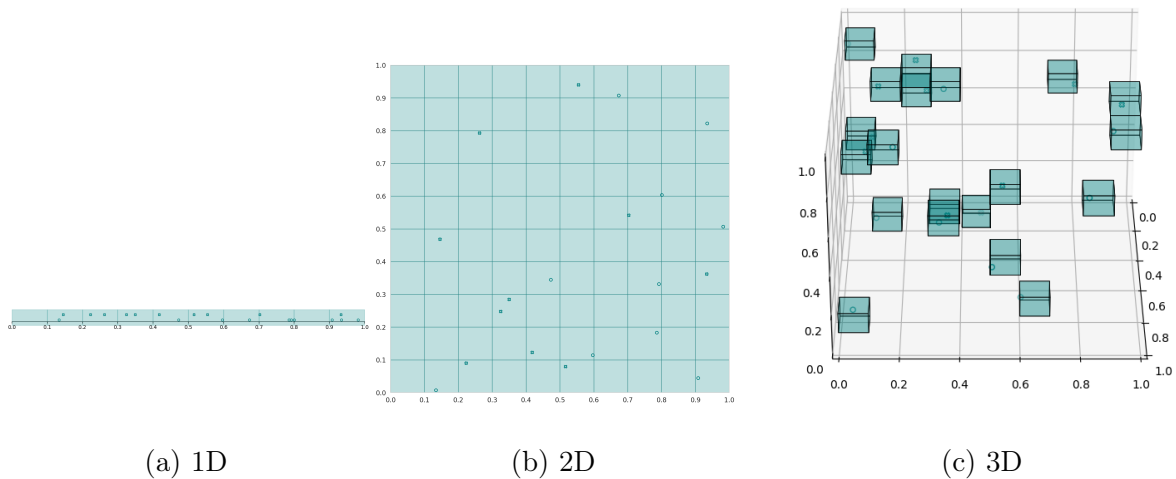


FIGURE 2.1.1 – Représentation de 20 points en 1D, 2D et 3D inspiré de [1]

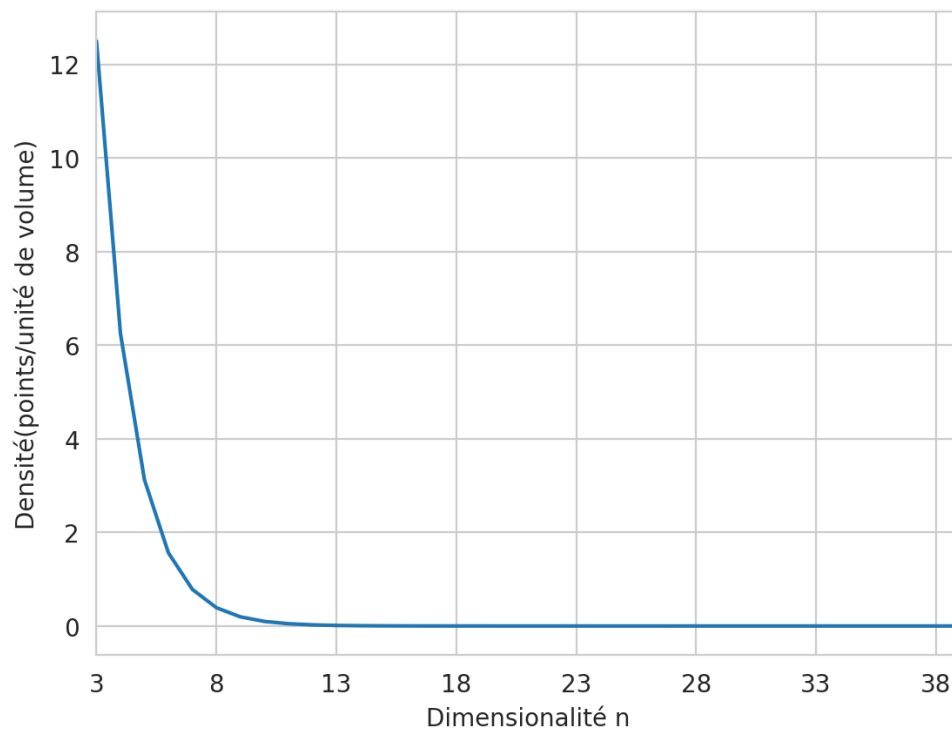


FIGURE 2.1.2 – Densité du nombre de points par unité de volume

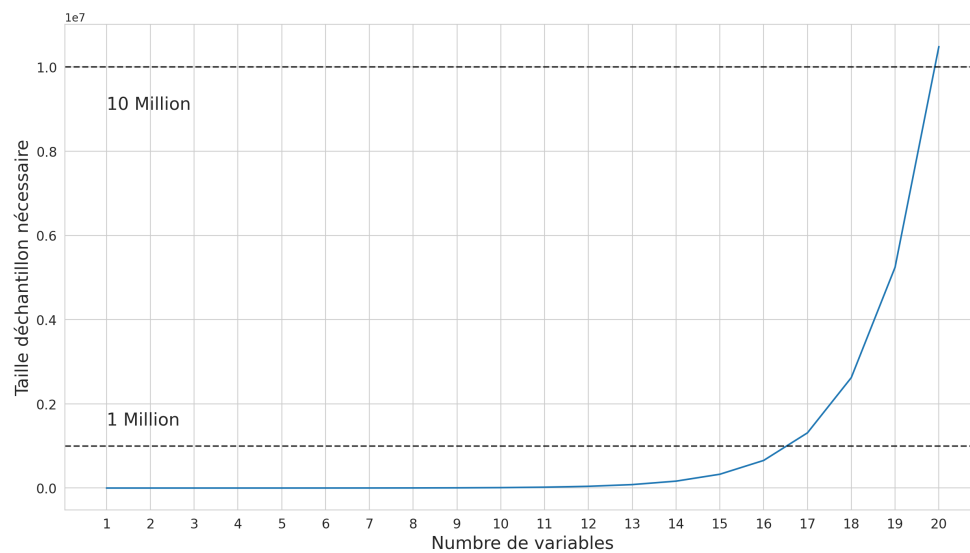


FIGURE 2.1.3 – Évolution de la taille d'échantillon nécessaire pour garder la même densité de points en fonction du nombre de variables

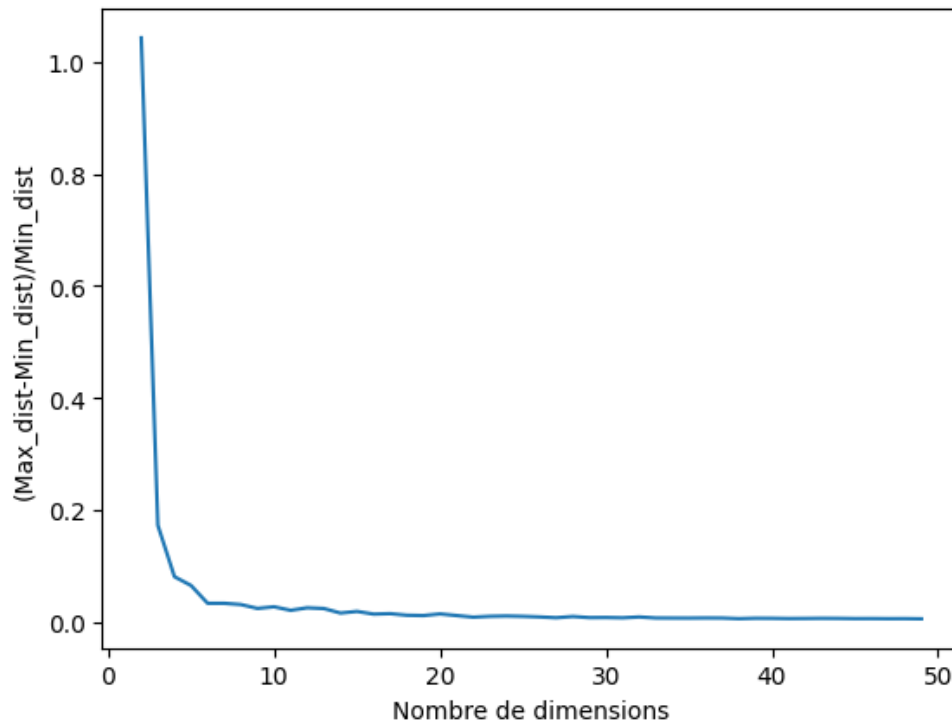


FIGURE 2.1.4 – Évaluation de l'écart entre la distance maximale et minimale des points en fonction du nombre de dimensions.

2.2 Réduction de la dimensionnalité comme solution pour lutter contre la dimensionnalité

Pour palier au problème de la malédiction de la dimension, une solution pourrait être d'augmenter la taille de l'échantillon d'entraînement de façon à atteindre la densité suffisante des observations, mais nous avons vu que cette tâche est difficile à réaliser dans la pratique.

Une autre solution à ce problème est la réduction de la dimensionnalité, que nous discuterons dans cette section.

La réduction de la dimensionnalité peut se faire de deux manières : La sélection des variables (choisir un sous-ensemble de variables pertinentes) et la projection dans un espace de dimension réduit. C'est cette dernière que nous présenterons dans cette section.

La projection est le processus de réduction du nombre de dimensions d'un ensemble de données à haute dimension qui permet de projeter les données dans un sous-espace de dimension inférieure [10]. Elle est justifiée par le fait que dans la plupart des problèmes réels, les observations ne sont pas uniformément réparties dans les différentes dimensions et certaines variables sont presque constantes ou fortement corrélées. Il existe une proximité ou un sous-espace de dimension inférieure qui contient tout ou presque tout le jeu d'entraînement. Cependant, la projection peut entraîner une perte d'information et ne pas représenter fidèlement les relations entre les données.

Il faut également noter que les données à haute dimension peuvent contenir des relations complexes et non linéaires qui ne sont pas facilement détectables par des méthodes de visualisation simples. Par conséquent, il est important de considérer ces difficultés lors du choix d'une méthode afin d'obtenir des résultats significatifs.

Nous allons dans cette partie présenter deux méthodes pour réduire la dimension : une méthode linéaire (PCA) et méthode non linéaire (t-SNE) tiré de l'article [11].

2.2.1 L'analyse en composantes principales

L'Analyse en Composantes Principales (ACP), ou PCA en anglais, est une méthode de calcul largement utilisée pour la compression de données et l'extraction de caractéristiques. Elle repose sur la transformation de Karhunen-Loève, également connue sous le nom de transformation de Hottelling [11]. Selon cet article, l'ACP permet de représenter un vecteur aléatoire dans un nouvel espace en utilisant les vecteurs propres de sa matrice de covariance. Contrairement à d'autres méthodes, l'ACP ne fait pas d'hypothèses sur les fonctions de densité de probabilité des données. Au lieu de cela, elle est entièrement basée sur les données elles-mêmes en utilisant uniquement la matrice de covariance. L'objectif principal de l'ACP est de réduire la redondance entre les variables composées par le vecteur aléatoire $\mathbf{x} \in \mathbb{R}^p$ où p est le nombre de variables. Cela est mesuré par les corrélations entre ces variables. L'ACP élimine alors les corrélations et permet ensuite de réduire la dimensionnalité en trouvant de nouvelles caractéristiques qui sont des combinaisons linéaires des caractéristiques d'origine. Étant donné qu'elle dépend uniquement de la matrice de covariance, l'ACP est une méthode statistique du second ordre. Elle peut se faire soit en maximisant la variance de la nouvelle représentation compacte Y , soit en minimisant l'erreur quadratique moyen entre les données d'origine X et leur représentation réduite Y .

Supposons $B = [U^T, R^T]$ une base orthogonale de $\mathbb{R}^p$ composé de deux vecteurs : U^T et R^T avec $U^T = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d]$ les d composantes dé-corrélées que nous souhaitons obtenir ($d < p$) ; $R^T = [\mathbf{u}_{d+1}, \mathbf{u}_{d+2}, \dots, \mathbf{u}_m]$ représente le sous espace des composantes à écarter. U définit ainsi le sous-espace linéaire dans lequel on projettera nos données. On a le problème suivant : étant donné un espace de dimension p , comment trouver le sous-espace de direction les $\mathbf{u}_j$, $j = 1, 2, \dots, d$, de sorte que si nous y projetons nos données d'origine, la variance des nouvelles données obtenues soit maximale. Nous cherchons donc à obtenir les $\mathbf{u}_j$ en gardant autant que possible la généralité de nos données d'origines. Supposons $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$ un échantillon centré avec $\mathbf{x} \in \mathbb{R}^p$. Dans la base U , $\mathbf{x}$ peut être réécrit comme :

$$\mathbf{x} = \sum_{j=1}^p (\mathbf{x}^T \mathbf{u}_j) \mathbf{u}_j = \sum_{j=1}^p w_j \mathbf{u}_j \quad (2.2.1)$$

où les w_j sont les coefficients d'expansion.

Nous obtenons donc un nouveau vecteur $\mathbf{y} \in \mathbb{R}^d$ en projetant $\mathbf{x}$ sur U par la transformation $\mathbf{y} = T\mathbf{x}$; on obtient :

$$\mathbf{y}^T = \mathbf{x}^T U^T = \sum_{j=1}^p w_j \mathbf{u}_j^T [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d]; \quad (2.2.2)$$

étant dans une base orthogonale, $\mathbf{u}_i^T \mathbf{u}_j = 1$ pour $i = j$ et $\mathbf{u}_i^T \mathbf{u}_j = 0$ pour $i \neq j$; ce qui conduit à :

$$\mathbf{y}^T = [w_1, w_2, \dots, w_d] \quad (2.2.3)$$

Cherchons donc à maximiser la fonction suivante : $V(U) = E[\|\mathbf{y}\|^2]$. On obtient :

$$V(U) = E[\mathbf{y}^T \mathbf{y}] = \sum_{j=1}^d E[w_j^2]. \quad (2.2.4)$$

Sachant que w_j est la projection de $\mathbf{x}$ sur $\mathbf{u}_j$, ie $w_j = \mathbf{x}^T \mathbf{u}_j$; On a :

$$V(U) = \sum_{j=1}^d E[\mathbf{u}_j^T \mathbf{x} \mathbf{x}^T \mathbf{u}_j] = \sum_{j=1}^d \mathbf{u}_j^T E[\mathbf{x} \mathbf{x}^T] \mathbf{u}_j = \sum_{j=1}^d \mathbf{u}_j^T \Sigma_x \mathbf{u}_j, \quad (2.2.5)$$

où Σ_x représente la matrice de covariance de X . Nous devons donc résoudre le problème d'optimisation suivant :

$$\arg \max_{\mathbf{u}_j} \sum_{j=1}^d \mathbf{u}_j^T \Sigma_x \mathbf{u}_j \quad \text{subject to} \quad \|\mathbf{u}_j\| = 1 \quad \text{for} \quad j = 1, 2, \dots, d \quad (2.2.6)$$

ce qui est résolu en utilisant le multiplicateur de Lagrange. La fonction de Lagrange est définie par :

$$V(U, \lambda_1, \lambda_2, \dots, \lambda_d) = \sum_{j=1}^d \mathbf{u}_j^T \Sigma_x \mathbf{u}_j - \sum_{j=1}^d \lambda_j (\mathbf{u}_j^T \mathbf{u}_j - 1) \quad (2.2.7)$$

La dérivée partielle sur $\mathbf{u}_j$ donne :

$$\frac{\partial}{\partial \mathbf{u}_j} V(U, \lambda_1, \lambda_2, \dots, \lambda_d) = 2\Sigma_x \mathbf{u}_j - 2\lambda_j \mathbf{u}_j = \Sigma_x \mathbf{u}_j - \lambda_j \mathbf{u}_j = 0, \text{ ce qui conduit à } \Sigma_x \mathbf{u}_j = \lambda_j \mathbf{u}_j \quad (2.2.8)$$

Nous pouvons donc réécrire notre problème d'optimisation de la manière suivante :

$$\arg \max_{\mathbf{u}_j} \sum_{j=1}^d \mathbf{u}_j^T \Sigma_x \mathbf{u}_j = \arg \max_{\mathbf{u}_j} \sum_{j=1}^d \mathbf{u}_j^T \lambda_j \mathbf{u}_j = \arg \max_{\mathbf{u}_j} \sum_{j=1}^d \lambda_j \quad (2.2.9)$$

L'ACP n'est performante que si les données présentent une structure linéaire. Cependant, si les données présentent des structures plus complexes qui sont des fonctions non linéaires des caractéristiques originales, l'ACP standard échouera à capturer des informations significatives. Heureusement, l'ACP avec noyau nous permet de généraliser l'ACP standard à une réduction de dimensionnalité non linéaire mais ne fais pas partie de notre présentation.

2.2.2 t-SNE

Parmi les techniques de visualisation existantes, on peut citer la projection multidimensionnelle (MDP), qui permet de projeter les données dans un espace de dimensions inférieures tout en préservant les relations entre les données. L'algorithme t-SNE (t-distributed stochastic neighbor embedding) est une technique de MDP populaire pour la visualisation des données creuses à haute dimension. Elle est basée sur son prédécesseur SNE.

Selon l'article [11], le SNE est basé sur la distance euclidienne entre les points $\mathbf{x}_i \in \mathbb{R}^p$ pour $i = 1, 2, \dots, n$ afin de définir les probabilités conditionnelles. La similarité entre deux points $\mathbf{x}_i$ et $\mathbf{x}_j$ est la probabilité conditionnelle $p_{j|i}$ selon laquelle $\mathbf{x}_i$ choisirait $\mathbf{x}_j$ comme voisin si la sélection se fait proportionnellement à une densité de probabilité sous une Gaussienne centrée en $\mathbf{x}_i$ définie par :

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}, \quad (2.2.10)$$

où σ_i^2 est la variance de la Gaussienne centrée sur $\mathbf{x}_i$. Cette probabilité prend des grandes valeurs pour des points assez proches et tend vers zéro pour des points éloignés. On définit de manière similaire une probabilité conditionnelle dans l'espace de faible dimension par $q_{j|i}$ tel que :

$$q_{j|i} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{y}_i - \mathbf{y}_k\|^2)}, \quad (2.2.11)$$

où $\mathbf{y}_i$ et $\mathbf{y}_j$ dans R^2 . Sachant que $\forall i, p_{i|i} = q_{i|i} = 0$, le but du SNE est de chercher un espace de faible dimension qui minimise la distance entre les deux probabilités. Une mesure statistique permettant d'évaluer la distance entre deux distributions de probabilités est la divergence de Kullback-Leibler encore appelé entropie relative. Le SNE minimise alors la somme de cette entropie pour tous les points en utilisant la méthode de la descente de gradient :

$$C = \sum_{i=1}^n KL(P_i || Q_i) = \sum_{i=1}^n \sum_{j=1}^n p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \quad (2.2.12)$$

où P_i représente la distribution de probabilité conditionnelle pour tous les points étant donné $\mathbf{x}_i$ dans l'espace à haute dimension et Q_i la probabilité conditionnelle pour tous les points étant donné $\mathbf{y}_i$ dans l'espace à faible dimension. Le SNE a donc pour but de conserver autant que possible la structure locale existante (dans l'espace à haute dimension) dans l'espace à faible dimension.

La variance σ_i^2 varie en fonction de la densité des régions. On cherche à attribuer les plus petites valeurs dans les zones denses que dans les zones creuses. On définit la mesure de perplexité comme étant :

$$\text{Perp}(P_i) = 2^{H(P_i)}$$

où $H(P_i)$ est l'entropie de Shannon définie par :

$$H(P_i) = - \sum_{j=1}^n p_{j|i} \log_2 p_{j|i}$$

Le SNE cherche les valeurs de σ_i^2 qui produisent les P_i avec une perplexité fixée par l'utilisateur. Cette dernière est interprétée comme une mesure de lissage du nombre de voisins effectif à chaque point généralement compris entre 5 et 50. La minimisation de la fonction objective par la descente de gradient revient à résoudre la question suivante jusqu'à convergence :

$$Y^{(t)} = Y^{(t-1)} - \eta \frac{\partial C}{\partial Y} + \alpha(t) (Y^{(t-1)} - Y^{(t-2)}), \quad (2.2.13)$$

où $Y^{(t)}$ représente l'ensemble de données obtenues à l'itération t , η est le taux d'apprentissage et $\alpha(t)$ est le paramètre qui accélère la convergence. Le t-SNE est obtenu en utilisant la distribution de Student à 1 degré de liberté où l'expression de q_{ij} est remplacée par :

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}} \quad (2.2.14)$$

dans l'espace à faible dimension.

L'algorithme t-SNE a une complexité de $O(n^2m + n^2h)$ où n , p et h représentent respectivement le nombre d'échantillons, le nombre de caractéristiques et le nombre d'itérations. L'algorithme t-SNE est considéré comme étant à la pointe de la réduction de dimension pour la visualisation de données. Malgré ses performances remarquables, la t-SNE présente également certaines limitations :

- les approches SNE et t-SNE ne préservent pas les interactions à longue distance entre les points de données et génèrent des visualisations dans lesquelles l'arrangement des groupes d'observations non voisins n'est pas informatif.
- le coût de calcul élevé, en particulier dû à l'optimisation numérique de sa fonction objective, le rend prohibitif pour les ensembles de données plus volumineux (il ne possède pas de solution analytique).
- Étant donné que l'algorithme est stochastique, plusieurs initialisations peuvent produire des résultats distincts.

La figure 2.2.1 montre clairement que l'ACP n'arrive pas à conserver la structure des données du fait de la structure non linéaire de celle-ci. La conséquence est que deux points éloignés dans l'espace à haute dimension se retrouvent proches dans l'espace à faible dimension (c'est le cas des points violets et jaune dans les graphes (a) et (b)). Le t-SNE quant à elle fait l'effort de préserver cette structure (graphe (c)) ; on voit bien que les points violets et jaunes sont séparés.

En somme chaque technique de réduction de la dimensionnalité a ses avantages et ses limites en termes de représentativité des données et de performance de visualisation. Il est donc important d'évaluer différentes techniques de visualisation pour les données creuses à haute dimension et de choisir celle qui convient le mieux à l'analyse des données à étudier.

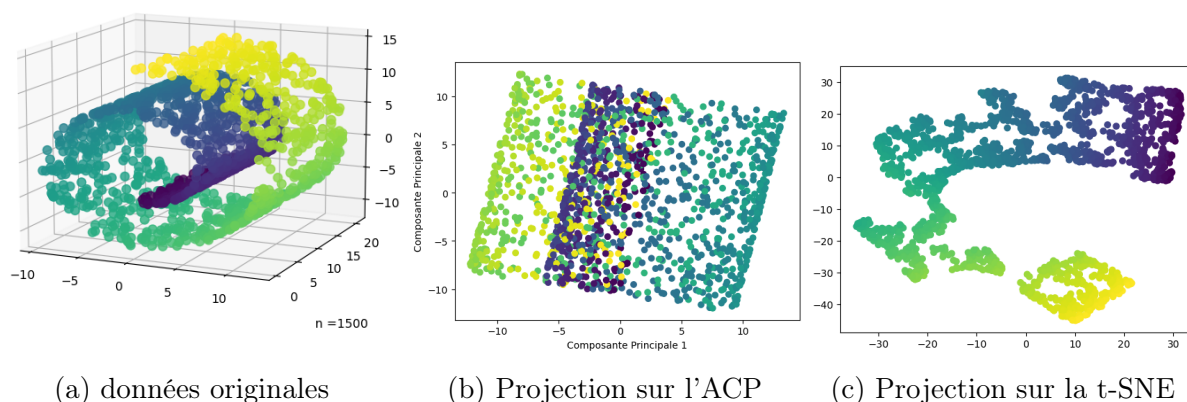


FIGURE 2.2.1 – Projection des données en utilisant les deux méthodes de réduction de dimension (inspiré du code python de Scikit-learn)

Dans la section suivante, nous allons présenter quelques travaux passés sur les méthodes de regroupements.

2.3 Quelques travaux passés sur le regroupement partiel des données

L'article [12] présente quelques travaux passés sur le regroupement des données à haute dimension

L'idée de départ est de réduire la dimensionnalité avant d'appliquer les algorithmes de regroupement. En effet, il s'agit d'effectuer une décomposition matricielle en approximant la matrice de données X de taille $n \times p$ où p est le nombre de variables quantitatives et n le nombre d'objets ou points dans l'espace $\mathbb{R}^p$ comme $X \approx AB$ où A est une matrice $n \times q$ et B est une matrice $q \times p$, ($q \leq p$). Ensuite, on peut regrouper les observations en utilisant A comme matrice de données, plutôt que X .

A ce sujet, plusieurs auteurs ont proposé d'utiliser le PCA pour obtenir une matrice A de dimension réduite, puis regrouper les n lignes de A . Cette approche a été largement utilisée dans la littérature sur les méthodes de regroupement en classes, remontant dans les années 80 [12]. Certaines études ont montré que la PCA ne permet pas forcément une bonne séparation entre les sous-groupes surtout avec les structures non linéaires. Chang dans [13] a démontré dans un article par simulations et également avec des données réelles que la pratique de réduire les données aux premières composantes principales avant le regroupement en classe n'est pas justifiée en général. Les composantes principales avec les valeurs propres les plus élevées ne permettent pas nécessairement une bonne séparation entre les sous-groupes, et que la prise d'un sous-ensemble de composantes principales peut entraîner une perte importante d'informations sur les groupes dans les données [13].

Des recherches plus récentes que celles présentées précédemment se sont appuyées sur un cadre de regroupement basé sur un modèle (model-based clustering framework). L'idée fondamentale est la suivante : Etant donné une matrice $X(n \times p)$, on peut modéliser les lignes de X ($\mathbf{x}_1 \dots \mathbf{x}_n$) comme observations multivariées indépendantes et identiquement distribuées tirées d'un modèle de mélange à K composantes. On utilise généralement un mélange de gaussiennes dont la densité $f(x_1 \dots x^n)$ est définie par :

$$f(\mathbf{x}_1, \dots, \mathbf{x}_n) \equiv \prod_{i=1}^n \sum_{k=1}^K \pi_k f_k(\mathbf{x}_i | \mu_k, \Sigma_k), \quad (2.3.1)$$

où $\pi_k (k = 1, \dots, K)$ est la probabilité qu'une observation appartienne au groupe k : $\pi_k > 0, \sum_{k=1}^K \pi_k = 1$, et

$$f_k(y_i | \mu_k, \Sigma_k) \equiv |2\pi\Sigma_k|^{-p/2} \exp\{-(\mathbf{x}_i - \mu_k)^T \times \Sigma_k^{-1}(\mathbf{x}_i - \mu_k)\} \quad (2.3.2)$$

est une densité normale multivariée avec pour moyenne μ_k et covariance Σ_k . La log-vraisemblance est donnée par :

$$l_{MIX}(\theta_1, \dots, \theta_K; \pi_1, \dots, \pi_K | \mathbf{x}_1, \dots, \mathbf{x}_n) \equiv \sum_{i=1}^n \log\left[\sum_{k=1}^K \pi_k f_k(\mathbf{x}_i | \mu_k, \Sigma_k)\right], \quad (2.3.3)$$

où $\theta_k = (\mu_k, \Sigma_k)$ et π_k sont des paramètres à déterminer en maximisant 2.3.4. L'algorithme EM (expectation-maximization algorithm) proposé par [14] permet d'aboutir à cette maximisation. Mais lorsque $p \approx n$ ou $p \gg n$, un problème se pose car la matrice de covariance Σ_k ($p \times p$) ne peut être estimée à partir de seulement n observations. Il faut donc trouver un espace de facteurs latents de faible dimension ; ce qui conduit une réduction de la dimensionnalité, mais pas à la parcimonie [12].

Dans le même article, l'auteur souligne que la classification basée sur un modèle se prête facilement à la sélection de variables. Au lieu de chercher μ_k et Σ_k qui maximisent la log-vraisemblance 2.3.4, on peut plutôt maximiser la log-vraisemblance en imposant une pénalité qui permet d'obtenir une parcimonie dans les caractéristiques. Par exemple, si l'on suppose que les p variables de X soient centrées pour avoir une moyenne nulle, alors [15] propose de maximiser la log-vraisemblance pénalisée,

$$l_{MIX}(\theta_1, \dots, \theta_K; \pi_1, \dots, \pi_K | x_1, \dots, x_n) \equiv \sum_{i=1}^n \log\left[\sum_{k=1}^K \pi_k f_k(x_i | \mu_k, \Sigma_k)\right] - \lambda \sum_{k=1}^K \sum_{j=1}^P |\mu_{kj}|, \quad (2.3.4)$$

où chaque Σ_k est considérée comme une matrice diagonale. Autrement dit, une pénalité L1 (ou lasso) est appliquée aux éléments de μ_k . Lorsque le paramètre de réglage non négatif λ est élevé, certains éléments de μ_k seront exactement égaux à zéro. Si, pour certaines variables j , $\mu_{kj} = 0$ pour tous les $k = 1, \dots, K$, alors la classification résultante ne tiendra pas compte de la caractéristique j . Ainsi, cela permet d'obtenir une classification parcimonieuse en termes de caractéristiques.

Une autre approche est présentée dans [12]. Il s'agit également une méthode de sélection de caractéristiques dans le cadre de la classification basée sur un modèle, en utilisant une approche entièrement différente. On transforme le problème de sélection de variables en un problème de sélection de modèle : des modèles contenant des sous-ensembles imbriqués (Interactions) de variables sont comparés. Les modèles imbriqués sont parcimonieux en termes de caractéristiques, et cela permet donc d'obtenir une méthode de classification parcimonieuse. Dans cette approche, les modèles sont sélectionnés en utilisant des critères d'information tels que le critère d'information bayésien (BIC) ou le critère d'information Akaike (AIC). En comparant les modèles contenant des sous-ensembles de variables différents, on peut identifier les sous-ensembles de variables qui maximisent le critère d'information, tout en minimisant le nombre de variables utilisées.

Cela permet d'obtenir une classification plus parcimonieuse, en n'utilisant que les variables les plus pertinentes pour la classification. Cette approche est particulièrement utile dans les situations où le nombre de variables est très élevé, car elle permet de réduire considérablement le nombre de variables utilisées sans sacrifier la qualité de la classification[16].

En somme, la réduction de dimension est une étape cruciale dans l'analyse des données à haute dimension. Les techniques telles que l'ACP et le t-SNE offrent des moyens efficaces de résoudre le problème de la malédiction de dimension en préservant les informations importantes tout en réduisant la complexité. En intégrant ces méthodes de réduction de dimension dans les processus d'analyse de données, nous pouvons améliorer notre compréhension des structures sous-jacentes, faciliter la visualisation et faciliter l'exploration des données complexes. Les approches traditionnelles de regroupement en classes, qui recommandent l'application du regroupement sur les sous-espaces obtenus après la réduction de dimension bien que critiqués par bon nombres d'auteurs donnent des résultats acceptables.

Dans le prochain chapitre, nous allons présenter la méthode K-means ainsi que ses variations tels que les méthodes Trimmed K-means, SK-means et RSKC.

Chapitre 3

Présentations du K-means et de ses variantes

Dans le chapitre précédent, nous avons constaté, conformément à la littérature, que les approches visant à réduire les dimensions ou à sélectionner les variables avant de procéder au regroupement des données ne fournissaient pas de résultats satisfaisants. Cela est dû au fait qu'il n'est pas garanti que l'espace réduit contienne l'ensemble du signal nécessaire pour identifier les différentes classes de manière précise.

C'est dans ce contexte que des méthodes de regroupement spécifiquement adaptées aux données parcimonieuses en haute dimension ont été développées. Dans cette section, nous allons examiner en détail quatre de ces méthodes de regroupement : le K-means de base, le Trimmed K-means, le Sparse K-means et le Robust and Sparse Kmeans Clustering.

Notre démarche débutera par une présentation du K-means de base, l'un des algorithmes de regroupement les plus répandus. Nous aborderons ses avantages ainsi que ses limites lorsqu'il est appliqué en haute dimension. Par la suite, nous exposerons les trois autres méthodes et procéderons à une comparaison entre elles en utilisant des ensembles de données simulées dans diverses configurations.

3.1 K-means

Le K-means est l'une des approches de regroupement les plus connues, qui vise à regrouper les données (objets) souvent en grande quantité en plusieurs classes liées sans aucune connaissance préalable des définitions de classes. Il a été premièrement proposé par MacQueen en 1967 [17]. L'objectif de cette approche est de regrouper ces objets de manière à ce que les objets d'un même groupe soient similaires autant que possible et différents des objets des autres groupes [3]. L'auteur de cet article souligne que le K-means est l'une des techniques de partitionnement avec une complexité presque linéaire. Autrement dit, l'algorithme de K-means nécessite moins d'espace de stockage par rapport à d'autres algorithmes car il n'a besoin de stocker que des objets et leurs centres. La capacité de stockage requise est de l'ordre de $O(n + K)$, où n représente le nombre d'objets ; de plus, il

a une relation linéaire avec le nombre d'objets, représenté par $O(iKn)$, où i est le nombre d'itérations de convergence. Étant donné que i est généralement petit et peut être borné, la plupart des changements se produisent dans les premières itérations [18]. En raison de sa simplicité, l'algorithme des K-means est l'un des algorithmes les plus utilisés dans le regroupement des objets. Il commence par l'initialisation aléatoire des centres des classes, ensuite on assigne chaque objet au centre le plus proche. L'algorithme est répété jusqu'à ce qu'un critère d'arrêt soit atteint (soit lorsque le nombre d'itérations maximal est atteint, soit lorsque les classes ne montrent plus aucun changement après un certain nombre d'itérations).

3.1.1 Algorithme de base

L'algorithme de K-means tiré de [10] se déroule de la manière suivante :

entrée : Considérons une matrice $n \times p$ où p est le nombre de variables quantitatives et n le nombre d'objets ou points dans l'espace $\mathbb{R}^p$.

sortie : Une partition des n objets en q classes $I_1, \dots, I_q$ tels que les classes $\{x_i : i \in I_k\}$, $k = 1, \dots, q$ sont plus ou moins homogènes.

Cet algorithme utilise une mesure de similarité qui est supposée additive et décomposée en p termes ; chaque terme dépendant d'une variable. On définit ainsi $d(x_i, x_{i'})$ comme la similarité entre les objets x_i et $x_{i'}$ comme : $d(x_i, x_{i'}) = \sum_{j=1}^p d_{ii'j}$; où $d_{ii'j}$ représente la similarité entre x_i et $x_{i'}$ pour la variable j .

La mesure de similarité la plus utilisée est la distance euclidienne entre deux points.

On distingue entre autre pour la j -ième variable :

- Le carré de la distance euclidienne, définie de la manière suivante : $d_{ii'j} = (x_{ij} - x_{i'j})^2$ pour .
- La distance de Manhattan, définie de la manière suivante : $d_{ii'j} = |x_{ij} - x_{i'j}|$.
- La distance de Minkowski qui est la généralisation des deux premières : $d_{ii'j} = |x_{ij} - x_{i'j}|^{1/a}$ où $a \geq 1$. Elle représente la distance de Manhattan si $a = 1$ et la distance euclidienne si $a = 2$.

L'algorithme de K-means se décrit à travers les étapes suivantes :

- (a) Sélectionner les centres des classes initiales (aléatoirement) $c_1^0, \dots, c_q^0 \in \mathbb{R}^p$. Cette étape peut également se faire en choisissant q classes aléatoirement et calculer leurs centres.
- (b) Assigner à chaque objet x_i le centre le plus proche de la manière suivante :

$$I_k^0 = \{i = 1, \dots, n : \|x_i - c_k^0\| < \|x_i - c_l^0\|, l \neq k\}.$$

où de manière générale $\|x_i - x_{i'}\|^2 = \sum_{j=1}^p (x_{ij} - x_{i'j})^2$. Nous obtenons ainsi une partition initiale $I_1^0, \dots, I_q^0$.

- (c) Calculer les centres des classes obtenus à l'itération précédente m de la manière suivante :

$$c_k^{m+1} = \frac{1}{|I_k^m|} \sum_{i \in I_k^m} x_i$$

où $|I_k^m|$ représente le nombre d'objets dans la classe k à l'itération m .

(d) Réassigner chaque point au centre le plus proche de la manière suivante :

$$I_k^{m+1} = \{i = 1, \dots, n : \|x_i - c_k^{m+1}\| < \|x_i - c_l^{m+1}\|, l \neq k\}.$$

Répéter les étapes (c) et (d) jusqu'à ce qu'il n'y ait plus de changement ou qu'un critère d'arrêt soit atteint.

3.1.2 La qualité de la partition

Considérons la moyenne empirique $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$. La somme totale des carrés (TSS) est donnée par l'expression suivante :

$$TSS = \sum_{i=1}^n \|x_i - \bar{x}\|^2 = \sum_{i=1}^n \sum_{j=1}^p (x_{ij} - \bar{x}_j)^2 = n \sum_{j=1}^p s_{jj},$$

où les s_{jj} ($j = 1, \dots, p$) représentent les éléments de la diagonale de la matrice de covariance. La TSS est donc proportionnelle à la trace de la matrice de covariance. En appliquant le théorème de König-Huygens suivant :

Theorem 3.1.1 (König-Huygens). Pour tout $a \in \mathbb{R}^p$,

$$\sum_{i=1}^n \|x_i - a\|^2 = \sum_{i=1}^n \|x_i - \bar{x}\|^2 + n\|\bar{x} - a\|^2$$

l'expression du TSS devient :

$$TSS = \sum_{k=1}^q \sum_{i \in I_k} \|x_i - \bar{x}\|^2 = \sum_{k=1}^q [\sum_{i \in I_k} \|x_i - \bar{x}_k\|^2 + |I_k| \|\bar{x}_k - \bar{x}\|^2]$$

d'où

$$TSS = \sum_{k=1}^q \sum_{i \in I_k} \|x_i - \bar{x}_k\|^2 + \sum_{k=1}^q |I_k| \|\bar{x}_k - \bar{x}\|^2$$

donc

$$TSS = W + B$$

où W est l'erreur quadratique intra classe et B l'erreur quadratique inter classe avec

$$W = \sum_{k=1}^q \sum_{i \in I_k} \|x_i - \bar{x}_k\|^2, B = \sum_{k=1}^q |I_k| \|\bar{x}_k - \bar{x}\|^2$$

La TSS étant constante, on peut choisir soit de minimiser W soit de maximiser B .

3.1.3 Choix des paramètres du K-means

Bien que K-means soit un algorithme simple et largement utilisé, il est nécessaire de faire 2 choix importants pour son déploiement. Le premier est le nombre de classes à fournir à l'algorithme et le second est le choix des centres initiaux. Ces 2 aspects influent sur la qualité de la partition résultante et nécessitent des considérations spécifiques pour l'amélioration des résultats [3].

Détermination du nombre de classes Le choix du nombre optimal de classes est une étape cruciale dans l'utilisation de K-means. Dans l'article [3], plusieurs méthodes différentes sont présentées ; parmi lesquelles ont a :

1. Méthode Elbow.

Cette méthode visuelle est la plus ancienne pour déterminer le nombre réel de classes dans un ensemble de données appelée la méthode du coude [19]. Selon cette article l'idée est de commencer avec $K = 2$, et d'augmenter cette valeur à chaque étape en évaluant les classes et le coût W associé à l'entraînement. À une certaine valeur de K , le coût diminue de manière significative, puis atteint un plateau lorsque on l'augmente davantage. Cette valeur représente la valeur de K recherchée.

La logique derrière cette méthode est qu'à un moment donné, lorsque on augmente le nombre de classes, la nouvelle classe se trouve très proche de certaines classes existantes. Par conséquent, notre objectif est d'obtenir une petite valeur de K qui a encore un coût minimal. Le coude représente généralement où il faut commencer. Bien évidemment le W vaut 0 lorsque K est synchronisé avec le nombre total d'observations, car chaque observation est sa propre classe et il n'y a pas d'erreur entre celle-ci et le centre de sa classe. Cependant, la méthode du coude n'est pas toujours claire et donne parfois plusieurs coudes et même parfois pas de coude pour certaines bases de données. Dans cette situation, il est conseillé d'utiliser une autre méthode qui pourrait éventuellement donner des meilleurs résultats.

2. Méthode de la Silhouette.

Cette méthode a été introduite par Kaufman et Rousseeuw dans les articles [20, 21]. La largeur de silhouette est une mesure utilisée pour évaluer la qualité d'un regroupement d'entités en classes. Elle prend en compte la distance entre chaque entité et les autres entités de sa propre classe, ainsi que la distance entre cette entité et les entités des autres classes. Cette mesure permet de déterminer si les entités sont bien regroupées ou non.

La largeur de la silhouette $s(i)$ pour une entité $i \in I$ est définie par :

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

où $a(i)$ est la distance moyenne entre i et toutes les autres entités de la même classe et $b(i)$ est le minimum des distances moyennes entre l'entité i et toutes les entités de chacun des autres classes.

La largeur de silhouette peut prendre des valeurs entre -1 et 1 . Si la valeur pour une entité est proche de zéro, cela signifie que cette entité pourrait être classée dans un autre cluster. Si la silhouette valeur est proche de -1 , cela signifie que l'entité est mal classée. En revanche, si toutes les valeurs sont proches de 1 , cela indique que le regroupement est de bonne qualité.

La qualité globale du regroupement peut être caractérisée par la moyenne des largeurs de silhouette de toutes les entités. En utilisant différentes valeurs pour le nombre de classes, on peut déterminer le nombre optimal de classes en choisissant celui qui maximise la largeur de silhouette moyenne. L'article [22] présente une étude comparative de différentes méthodes pour évaluer la qualité des classes, y compris la méthode de la largeur de silhouette. Les auteurs soulignent que la méthode de la largeur de silhouette peut ne pas être suffisamment sensible pour détecter des classes de formes complexes. Les auteurs concluent que l'utilisation de plusieurs méthodes pour évaluer la qualité des classes est recommandée pour obtenir une évaluation plus complète et précise.

3. **Méthode de la statistique Gap.** Les auteurs de l'article [3] ont proposé une méthode pour estimer le nombre optimal de classes dans un ensemble de données qui peut être utilisée avec n'importe quel algorithme de regroupement. Cette méthode fonctionne en comparant le changement de dispersion intra-cluster W_k à mesure que le nombre K de classes augmente, et en identifiant visuellement un coude dans le graphique résultant pour déterminer le nombre optimal de classes. D'autre part, La statistique d'écart permet de normaliser le graphique $\log(W_k)$ en utilisant une distribution de référence nulle des données à des fins de comparaison et de sélectionner le plus petit K comme valeur réelle, qui maximise $\text{Gap}(K)$. Pour plus de détails voir [23].

Les étapes pour la statistique de Gap [3] :

- (a) Créer des classes en utilisant une plage de valeurs de k avec $K = 1, \dots, K_{max}$ et calculer W_k pour chaque valeur.
- (b) Créez B ensembles de données de référence en utilisant l'une des méthodes mentionnées dans [23]. Regroupez-les avec une gamme de $K = 1, \dots, K_{max}$ et calculez les statistiques d'écart prédites.

$$\text{Gap}(k) = \left(\frac{1}{B}\right) \sum_{b=1}^B \log W_{kb}^* - \log W_k,$$

où W_{kb}^* représente la fonction objective du b^e ensemble de données et $\left(\frac{1}{B}\right) \sum_{b=1}^B \log W_{kb}^*$ la moyenne des B ensembles.

- (c) Maintenant avec $\bar{w} = \left(\frac{1}{B}\right) \sum_{b=1}^B \log W_{kb}^*$, calculer l'écart type

$$sd(k) = \left[\left(\frac{1}{B}\right) \sum_{b=1}^B \log W_{kb}^* - \bar{w}\right]^{\frac{1}{2}}$$

et on définit $s_k = \sqrt{1 + \frac{1}{B}}sd(k)$

- (d) Choisir la valeur plus petite valeur de K tels que $\text{Gap}(k) \geq \text{Gap}(k)(k+1) - s_k + 1$

Choix des centres initiaux Il existe plusieurs méthodes d'initialisation des centres de classes dans l'algorithme de K-means. Certaines de ces méthodes consistent à sélectionner des centres de classes de manière aléatoire, tandis que d'autres utilisent des techniques plus sophistiquées pour trouver ces centres de cluster. Nous présentons 3 parmi celles proposées par l'article [18].

1. La première approche optimise l'algorithme original des K-means en proposant une méthode pour choisir les classes initiales. Au lieu de choisir les centres initiaux de manière aléatoire, on divise chaque dimension de données d'entrée en k segments, où k est le nombre souhaité de classes (voir l'article initial [24] pour plus de détails). Après la division de l'espace de données en cellules, la fréquence de chaque cellule est calculée et les k cellules ayant la plus grande fréquence de données sont choisies pour représenter les classes initiales. Si certaines cellules ont la même fréquence, les cellules adjacentes ayant la même fréquence la plus faible sont fusionnées jusqu'à ce que nous obtenions k cellules distinctes. Ensuite, les centres initiaux sont calculés en prenant la moyenne des points de données dans les segments respectifs pour obtenir les k centres initiaux. Cette méthode peut être plus efficace que la méthode de sélection aléatoire des centres initiaux, car elle permet de choisir des centres qui sont plus représentatifs de l'ensemble de données.
2. Une autre méthode permet de déterminer les centres initiaux afin de produire les classes avec bonne précision. Elle utilise la distance euclidienne pour calculer la distance entre deux points et se déroule en plusieurs étapes :
 - (a) Tout d'abord, elle calcule la distance entre chaque point et tous les autres points dans l'ensemble de données.
 - (b) Elle trouve ensuite la paire de points la plus proche l'une de l'autre et crée un premier ensemble A_1 avec ces deux points. Ces deux points sont ensuite supprimés de l'ensemble de données initial.
 - (c) Elle détermine alors le point de données le plus proche de l'ensemble A_1 et l'ajoute à A_1 , puis le supprime de l'ensemble de données initial.
 - (d) Elle répète ce processus jusqu'à ce que le nombre d'éléments dans l'ensemble A_1 atteigne un seuil prédéfini.
 - (e) Elle forme ensuite un nouvel ensemble de données A_2 et répète les étapes b à d jusqu'à ce que K ensembles de données soient obtenus.
 - (f) Enfin, les centres initiaux sont obtenus en faisant la moyenne de tous les vecteurs dans chaque ensemble de données.

3.1.4 Quelques inconvénients du K-means

Bien que le K-means ait plusieurs avantages tels que sa rapidité d'exécution, sa facilité à mettre en oeuvre et sa complexité linéaire, il présente néanmoins quelques limites [25].

- Cette méthode requiert la spécification préalable du nombre de centres de classes. Étant donné que les centres initiaux sont sélectionnés de manière aléatoire dans l'algorithme de base, il est possible d'obtenir des solutions différentes à chaque exécution de l'algorithme. Afin de résoudre ce problème, nous avons proposé plusieurs méthodes d'initialisation des centres.
- Cet algorithme converge souvent vers un minimum (maximum) local si un mauvais choix des centres initiaux a été fait.
- Cet algorithme est très sensible à la présence des valeurs aberrantes.
- Cette méthode s'applique uniquement lorsque la moyenne est définie, elle ne s'applique donc pas aux données catégorielles.

3.2 Trimmed K-means

Les auteurs de l'article [26] ont proposé une modification de l'algorithme de K-means standard dans le but d'obtenir un regroupement robuste aux valeurs aberrantes. Il s'agit simplement de remplacer l'étape (b) de l'algorithme de base par l'étape (b') ci-dessous :

- (b') Lorsqu'une attribution de regroupement est donnée, éliminez $\alpha\%$ des observations ayant la plus grande distance par rapport à leurs centres de regroupement, puis mettez à jour les centres de regroupement en utilisant la moyenne des échantillons restants dans chaque groupe.

Cela permet d'écarter les éventuelles valeurs aberrantes avant la mise à jour des centres à chaque itération. Ces valeurs seront réintégrées aux centres les plus proches à la fin du processus. Bien que cette approche soit robuste aux valeurs aberrantes, Elle reste limitée si les variables n'ont pas les mêmes poids. Elle hérite également des autres difficultés que rencontre le K-means. Si certains auteurs suggèrent d'utiliser des techniques de réduction de dimensionnalité comme l'ACP ou la sélection des variables pour résoudre ces problèmes, nous avons vu au premier chapitre que certains auteurs trouvent ces approches inappropriées. Il faut donc se tourner vers d'autres méthodes plus appropriées telles que des méthodes de regroupement parcimonieux.

3.3 Méthodes de regroupement parcimonieux

3.3.1 Motivations

Commençons par donner nos motivations et planter le décor avec un cadre général de groupement des données parcimonieux. Nous nous basons sur les graphiques proposés par l'article[2] dans cette partie. Il s'agit d'une simulation de données de 400 observations sur trois dimensions a, b, c normalement distribuées. Ces données sont divisées en 4 classes. Les

dimensions a, b ont respectivement pour moyennes 0.5 et -0.5 et pour écart type 0.2 et la variable c est bruit ($\mu = 0, \sigma = 1$). La figure 3.3.1 est une représentation graphique de ce jeu de données. On voit à travers ce graphique que les dimensions a et b permettent de distinguer uniquement les classes bleues et violettes tandis que les dimensions b et c permettent de distinguer les classes rouges et vertes. Pour mieux visualiser cela, regardons ce qui se passe dans chacune des dimensions, puis en combinant les dimensions deux à deux.

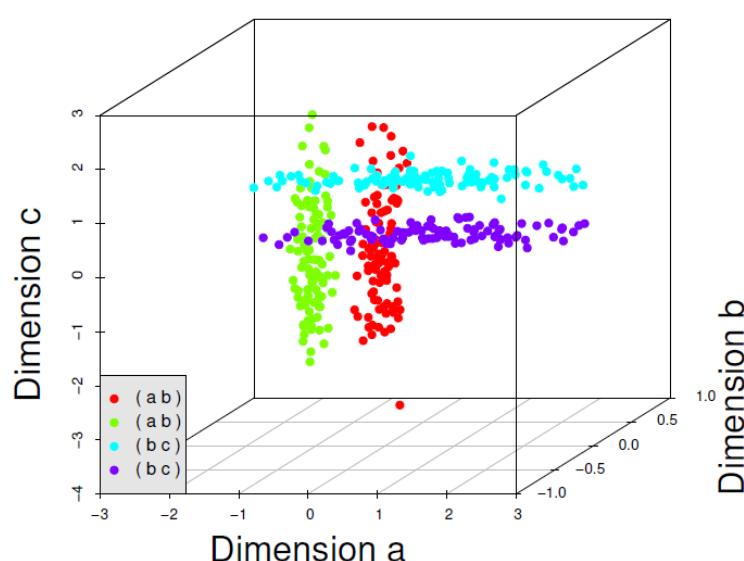


FIGURE 3.3.1 – Échantillon de données avec 4 classes (source [2])

Dans le graphe (a) de la figure 3.3.2, on voit que si on fait une projection sur la dimension a , les classes rouges et vertes sont clairement identifiées tandis que les classes bleues et violettes sont quasiment confondues. La dimension b permet de distinguer deux classes au lieu de quatre et les classes rouges et vertes sont confondues tandis que les classes bleues et violettes sont confondues. La dimension c ne permet pas de faire une distinction de classes.

La figure 3.3.3 montre que les dimensions a et b permettent de distinguer les classes vertes et rouges, ce qui signifie que ces classes sont créées par ces dimensions ; les deux autres sont confondues (graphe (a)). Ajouter la dimension c ne change rien car c'est du bruit. Les dimension b et c permettent de distinguer les classes bleues et violettes, les autres sont confondues (graphe (b)). En enlevant la dimension c on a le même résultat que dans la figure 3.3.2. Les dimensions a et c quant à elles ne permettent pas de différencier clairement les classes.

A partir cette illustration, nous constatons qu'aucune des dimensions seule ne permet de distinguer les quatre classes de départ et Même en les combinant deux à deux, on n'obtient pas le résultat escompté. Il est donc important de chercher pour chaque classe, quel est le

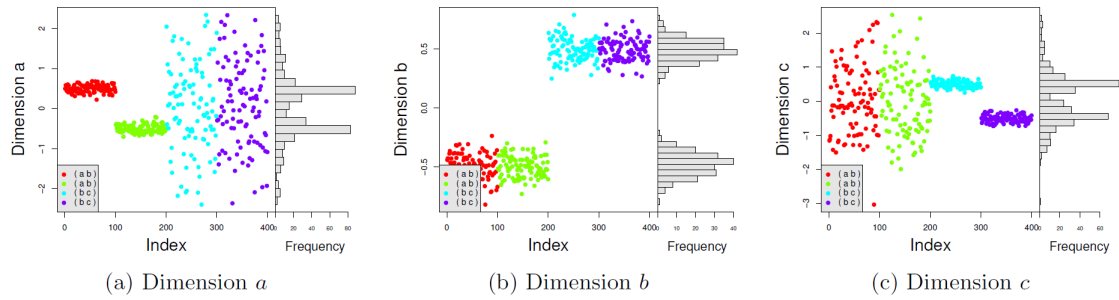


FIGURE 3.3.2 – Échantillon de données projetées sur une dimension avec histogramme (tiré de [2])

sous espace qui a contribué à sa construction. En d'autres termes, dans un espace à haute dimension, Il faut trouver les sous-ensembles de variables qui permettent de différencier les différentes classes.

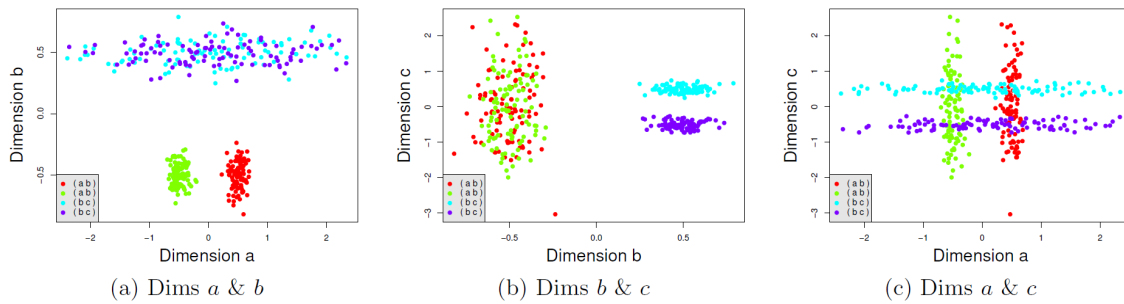


FIGURE 3.3.3 – Échantillon de données projetées sur deux dimensions (source [2])

3.3.2 Cadre théorique général

Comme présenté dans l'article [12], supposons toujours notre matrice X de dimension $n \times p$ et $X_j \in \mathbb{R}^n$ la variable j . Un bon nombre de méthodes de regroupements cherchent à résoudre le problème d'optimisation suivant :

$$\text{maximiser}_{\Theta \in D} \left\{ \sum_{j=1}^p f_j(X_j, \Theta) \right\}, \quad (3.3.1)$$

où $f_j(X_j, \Theta)$ est une fonction qui implique uniquement la j -ème variable et Θ un paramètre limité à l'ensemble D . Dans le cas du K-means, f_j est la dispersion entre les classes pour la variable j (Between cluster sum of squares) et Θ est la partition des observations dans K classes disjoints.

Une version parcimonieuse de (3.3.1) est alors donnée par :

$$\text{maximiser}_{\Theta \in D} \left\{ \sum_{j=1}^p w_j f_j(X_j, \Theta) \right\} \quad (3.3.2)$$

avec pour conditions : $\|w\|^2 \leq 1$ ($\sum_{j=1}^p w_j^2 \leq 1$), $\|w\|_1 \leq s$ ($\sum_{j=1}^p w_j \leq s$), $w_j \geq 0 \forall j$, où w_j est le poids correspondant à la variable j et s un paramètre tel que $1 \leq s \leq \sqrt{p}$.

L'idée de base est d'assigner des poids à chaque variable et se baser sur une mesure de la dissimilarité pondérée pour déterminer les classes. Quelques observations peuvent se faire sur l'expression (3.3.2) :

- (a) Si $w_1 = \dots = w_p$ dans (3.3.2), alors le critère se réduit à (3.3.1).
- (b) La pénalité L_1 , ou Lasso, sur w entraîne une parcimonie pour de petites valeurs du paramètre de réglage s : c'est-à-dire que certains des w_j seront égaux à zéro et peu de variables contribueront à la formation des classes. La pénalité L_2 joue également un rôle important, car sans elle, au plus un élément de w serait généralement non nul. Un critère impliquant un objectif linéaire soumis à la fois à une contrainte L_1 et L_2 peut également être utilisé. Pour plus de détail, voir [27].
- (c) La valeur de w_j peut être interprétée comme la contribution de la variable j à la classification parcimonieuse résultante. Une grande valeur de w_j indique une grande contribution de la variable j dans la classification et $w_j = 0$ signifie que la variable j n'est pas impliquée dans la classification.
- (d) En général, pour que l'expression (3.3.2) aboutisse à un regroupement parcimonieux non trivial, il est nécessaire que $f_j(X_j, \Theta) > 0$ pour une partie ou tout j car si $f_j(X_j, \Theta) < 0$ alors $w_j = 0$.

Un algorithme itératif pour optimiser l'équation (3.3.2) est utilisé : en fixant w , nous optimisons (3.3.2) par rapport à Θ , puis en fixant Θ , nous optimisons (3.3.2) par rapport à w . En général, on ne parvient pas à atteindre un optimum global de (3.3.2) avec cette approche itérative ; cependant, chaque itération garantit une augmentation de la fonction objectif. La première optimisation implique généralement l'application d'une procédure de regroupement standard à une version pondérée des données. Pour la seconde, étant donné Θ fixé, le problème (3.3.2) se réécrit de la manière suivante :

$$\text{maximiser}_w \{w^T a\} \quad (3.3.3)$$

avec pour conditions : $\|w\|^2 \leq 1$, $\|w\|_1 \leq s$ et $w_j \geq 0 \forall j$ où $a_j = f_j(X_j, \Theta)$. Pour plus de détails voir [28].

3.3.3 L'approche Sparse K-means (SK-means)

Dans cette section, nous allons présenter une version de K-means en utilisant (3.3.3) et en considérant unique la pénalisation de type Lasso.

Nous avons vu que le K-means standard donne une partition des observations en K classes disjoints en minimisant la dispersion intra-classes W donnée par l'expression :

$$W = \sum_{k=1}^q \sum_{i \in I_k} \|x_i - \bar{x}_k\|^2.$$

Cette expression peut également s'écrire sous la forme :

$$W = \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} \sum_{j=1}^p d_{i,i',j}, \quad (3.3.4)$$

où $d_{i,i',j}$ est la distance euclidienne des observations i et j le long de la dimension j et n_k est le nombre d'observations de la k ème classe. On peut soit minimiser cette expression soit maximiser la variance inter-classes (Between-cluster sum of squares) comme nous l'avons vu précédemment. Cette expression s'écrit comme :

$$B = \sum_{j=1}^p \left(\frac{1}{n} \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j} - \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} d_{i,i',j} \right), \quad (3.3.5)$$

où $\frac{1}{n} \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j}$ est la TSS de la j^e variable.

L'approche présentée dans l'article [12] propose une version pondérée de cette expression en ajoutant les poids w_j pour chaque variable j . Ainsi il faut résoudre le problème d'optimisation suivant :

$$\text{maximiser}_{I_1, \dots, I_k, \mathbf{w}} \left\{ \sum_{j=1}^p w_j \left(\frac{1}{n} \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j} - \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} d_{i,i',j} \right) \right\}, \quad (3.3.6)$$

avec pour conditions : $\|w\|^2 \leq 1$, $\|w\|_1 \leq s$ et $w_j \geq 0 \forall j$. Posons $a_j = \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j} - \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} d_{i,i',j}$ la variance inter-classe de la j^e variable. Remarquons que si les poids sont identiques à toutes les dimensions, c-à-d $w_1 = \dots = w_p$, on revient à un K-means standard et par identification à l'équation (3.3.2), $\Theta = I_1, \dots, I_k$, $f_j(X_j, \Theta) = \frac{1}{n} \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j} - \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} d_{i,i',j}$ et D est l'ensemble de toutes les partitions possibles des observations dans les K classes.

L'optimisation de (3.3.6) se fait en deux étapes principales :

1. Premièrement, garder les poids $w_1, \dots, w_p$ fixes et déterminer les classes en utilisant cette fois les distances pondérées.
2. Deuxièmement utiliser les classes $I_1, \dots, I_k$ trouvées précédemment considérées comme fixes et assigner à chaque variable un poids basé sur B . Les variables avec des larges B auront des poids élevés.

L'algorithme se fait de manière itérative de la manière suivante :

Algorithme de SK-means.

1. Initialiser les poids $\mathbf{w} : w_1 = \dots = w_p = \frac{1}{\sqrt{p}}$.
2. Exécuter les étapes suivantes jusqu'à convergence.
 - (a) Étant donné le vecteur de poids $\mathbf{w}$ et les centres $c_1, \dots, c_k$, résoudre le problème d'optimisation suivant :

$$\text{minimiser}_{I_1, \dots, I_k} \left\{ \sum_{k=1}^q \sum_{i \in I_k} \sum_{j=1}^p w_j (x_{i,j} - c_{k,j})^2 \right\},$$

où $c_{k,j}$ est le centre de la classe k pour la j^e variable. Ce qui permet d'assigner chaque point au centre le plus proche en utilisant la distance euclidienne pondérée (notons qu'à ce niveau, les poids sont identiques).

- (b) Étant donné le vecteur des poids $\mathbf{w}$ et les centres $I_1, \dots, I_k$, mettre à jour les centres en utilisant la moyenne pondérée des observations de chaque classe.
 - (c) Étant donné les classes $I_1, \dots, I_k$ et leurs centres respectifs, optimiser (3.3.6) suivant $\mathbf{w}$. Il s'agit de mettre à jour les poids en utilisant la proposition (1) : avec $a_j = \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j} - \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} d_{i,i',j}$ et $\delta = 0$ si $\|\mathbf{w}\|_1 < s$; sinon $\delta > 0$ est choisi tel que $\|\mathbf{w}\|_1 = s$.
3. On obtient ainsi les classes $I_1, \dots, I_k$ et les variables pondérées qui ont défini ces classes.

Proposition 1. La solution du problème de convexe 3.3.3 est $\mathbf{w} = \frac{S(a_+, \delta)}{\|S(a_+, \delta)\|_2}$ où a_+ est la partie positive de a et $\delta = 0$ si $\|\mathbf{w}\|_1 < s$; sinon $\delta > 0$ est choisi tel que $\|\mathbf{w}\|_1 = s$. S est l'opérateur de seuillage souple défini comme $S(a, c) = \text{signe}(a)(|a| - c)_+$. En assumant qu'il y a un unique vecteur maximal $\mathbf{a}$ et que $1 \leq s \leq \sqrt{p}$.

L'étape 2 de l'algorithme se répète jusqu'à ce que le critère d'arrêt :

$$\frac{\sum_{j=1}^p |w_j^t - w_j^{t-1}|}{\sum_{j=1}^p |w_j^{t-1}|} < 10^{-4}$$

soit satisfait ; où $\mathbf{w}^t$ est l'ensemble des poids obtenus à l'itération t . Concrètement, ce critère de convergence fonctionne de la manière suivante :

- À l'itération $t - 1$, les centres des classes et les poids associés à chaque dimension sont déterminés.
- À l'itération t , les nouveaux centres des classes sont calculés en fonction des points de données assignés à chaque classe. Les poids des dimensions sont également mis à jour en fonction.
- Pour chaque dimension j , calculez la différence entre le poids de cette dimension à l'itération t (représenté par w_j^t) et le poids de la même dimension à l'itération précédente $t - 1$ (représenté par w_j^{t-1}).
- Calculer le quotient $\frac{\sum_{j=1}^p |w_j^t - w_j^{t-1}|}{\sum_{j=1}^p |w_j^{t-1}|}$.

- Comparez le résultat de ce quotient avec le seuil fixé de 10^{-4} . Si le rapport est inférieur à ce seuil, cela signifie que les poids des dimensions ont convergé à un point où le changement entre les itérations successives est suffisamment petit.
- Si le critère de convergence est satisfait, l'algorithme s'arrête, car les poids des dimensions ont convergé et ne changent que de manière négligeable entre les itérations. Sinon, l'algorithme continue avec une nouvelle itération.

Ce critère de convergence vise à vérifier si les poids des dimensions convergent rapidement vers une valeur stable, indiquant ainsi que l'algorithme a atteint un état de convergence où les changements dans les poids des dimensions sont minimes. Cela permet d'économiser des itérations supplémentaires et de rendre l'algorithme plus efficace.

Sélection du paramètre de parcimonie s . Comme nous l'avons vu, l'algorithme de regroupement SK-means possède un paramètre d'ajustement, noté s , qui agit comme une borne L_1 sur le paramètre w dans l'équation (3.3.6). Il est important de noter que le nombre de classes, K , est supposé prédéterminé dans ce contexte. Il n'est pas simple de sélectionner s uniquement en se basant sur la maximisation de la fonction objective dans l'équation (3.3.6), car son augmentation entraînera également une augmentation de cette fonction. À la place, l'auteur utilise une approche basée sur les permutations qui présentent des similitudes avec la statistique de Gap utilisée dans la sélection du nombre optimal de classes dans le contexte du K-means.

Algorithme de sélection du paramètre s

1. Générer B ensembles de données $X_1, \dots, X_B$ en permutant les observations dans chaque variable indépendamment.
2. Pour chaque valeur indépendante du paramètre s :
 - (a) Calculer $O(s) = \sum_{j=1}^P w_i (\frac{1}{n} \sum_{i=1}^n \sum_{i'=1}^n d_{i,i',j} - \sum_{k=1}^q \frac{1}{n_k} \sum_{ii' \in I_k} d_{i,i',j})$, la fonction objective du SK-means avec le paramètre s sur l'ensemble de données X .
 - (b) Pour $b = 1, \dots, B$, calculer $O_b(s)$ la fonction objective du SK-means avec le paramètre s sur l'ensemble de données X_b .
 - (c) Calculer $\text{Gap}(k) = \log(O(s)) - (\frac{1}{B}) \sum_{b=1}^B \log(O_b(s))$.
3. Choisir s^* correspondant à la valeur du gap maximal.

Il est important de noter que dans les données originales, il peut exister des corrélations significatives entre les variables. Cependant, les variables des ensembles de données permutés $X_1, \dots, X_B$ sont décorréliées les unes des autres et sont supposées ne pas contenir des sous-groupes. La statistique de Gap mesure l'écart des fonctions objectives obtenues du regroupement obtenu sur les données réelles par rapport au regroupement obtenu sur des données issues des permutations. La valeur optimale du paramètre s s'obtient lorsque cet écart est maximal.

L'approche SK-means souffre des mêmes problèmes que le K-means standard en ce qui concerne le choix du nombre de classes au départ et l'initialisation des centres initiaux. Les mêmes techniques utilisées dans le K-means standard permettent de gérer ces problèmes.

Mais au lieu d'attribuer les mêmes poids à toutes les variables, cette approche à le mérite d'identifier les variables les plus pertinentes qui permettent de distinguer les classes. Cependant la sensibilité aux valeurs aberrantes restent d'actualité et il faut trouver une approche supplémentaire permettant de palier à ce problème.

3.3.4 L'approche Robust and Sparse K-mean Clustering (RSKC)

Cette approche a été présentée dans l'article [29]. C'est une version robuste du SK-means qui a pour but de gérer les valeurs aberrantes. L'étape 2(a) de l'algorithme précédent peut être détaillée de la manière suivante :

- (a) Étant donné le vecteur de poids $\mathbf{w}$, et les centres $c_1, \dots, c_q$, assigner les points aux centres les plus proches en utilisant la distance pondérée.
- (a') Mettre à jour les centres de classes en utilisant les moyennes pondérées

Une approche naïve pour rendre robuste le SK-means est de remplacer l'étape (a') par (b') dans le SK-means où les a_j sont calculés sans observations supprimées supposées être des valeurs aberrantes. l'auteur [29] souligne que cette approche présente un problème car si une observation est une valeur aberrante dans une variable non pertinente, elle peut ne pas être supprimé dans le processus du fait de son influence dans le calcul des centres. Cette variable se verrait attribuer un poids élevé à l'étape (c) parce que cette variable sera associée à un grand a_j

- (b') Étant donné les classes, supprimer les $\alpha\%$ d'observations ayant la plus grande distance de leurs centres respectifs et mettre à jour les centres de classes en utilisant les moyennes pondérées à partir des observations restantes dans chaque classe.

Pour résoudre ce problème, l'article propose une version améliorée : RSKC qui utilise le même cadre mais qui supprime temporairement les valeurs aberrantes dans le processus, une fois les classes identifiées en utilisant le SK-means, ces valeurs aberrantes sont ré-insérées aux classes les plus proches. L'algorithme peut être décrite par les étapes suivantes :

Algorithme de RSKC

- (a) Itérer les deux étapes suivantes jusqu'à convergence :
 - (i) Étant donné le vecteur de poids $\mathbf{w}$, et les centres $c_1, \dots, c_q$, résoudre le problème d'optimisation suivant :

$$\text{minimiser}_{I_1, \dots, I_k} \left\{ \sum_{k=1}^q \sum_{i \in I_k} \sum_{j=1}^p w_j (x_{ij} - c_{kj})^2 \right\},$$

où $c_{k,j}$ est le centre de la classe k pour la j^e variable. Ce qui permet d'assigner chaque point au centre le plus proche en utilisant la distance euclidienne pondérée.

- (ii) Étant donné les classes, supprimer les $\alpha\%$ d'observations ayant la plus grande distance de leurs centres respectifs et mettre à jour les centres de classes en utilisant les moyennes pondérées à partir des observations restantes dans chaque classe.

- (b) On note O_W l'ensemble de points supprimés à l'étape précédente que nous considérons comme premières valeurs aberrantes. Utilisons la partition obtenue pour calculer à nouveau les centres sans pondération que nous notons c'_k ($k = 1, \dots, q$). Pour chaque observation $\mathbf{x}_i$, calculons la distance non pondérée à son centre notée $d'_i = \|\mathbf{x}_i - c'_k\|^2$ où $i \in I_k$ et écartons à nouveau $\alpha\%$ des observations ayant la plus grande distance de leurs centres respectifs que nous notons O_E . On obtient une partition sans les observations des ensembles O_W et O_E .
- (c) Formons l'ensemble $O = O_W \cup O_E$ de tous les points mis à l'écart.
- (d) A partir de la partition obtenue à l'étape (b), et l'ensemble O , trouver le nouveau vecteur de poids $\mathbf{w}$ en résolvant le problème d'optimisation 3.3.6 où les a_j sont calculés sans les observations de l'ensemble O . Cela permet également d'assigner chaque observation de l'ensemble O à la classe la plus proche du regroupement final.

Le RSKC a trois paramètres : le paramètre de parcimonie s , la proportion α des points à écarter temporairement de l'analyse et le nombre de classes k . Le choix de s dépend du degré de parcimonie que l'on souhaite avoir dans la sélection des variables pertinentes ; le paramètre K pose le même problème que les deux algorithmes précédents. Selon l'article, $\alpha = 0.1$ est très souvent choisi dans la pratique. Mais il convient de tester plusieurs combinaisons de ces paramètres afin de choisir ceux qui donnent les meilleurs résultats.

En plus d'être robuste aux valeurs aberrantes, un autre avantage de cette méthode est qu'elle fonctionne avec les valeurs manquantes. Ce qui n'est pas le cas avec les deux autres approches.

Gestion des valeurs manquantes Soit l'observation $x_i = (x_{i1}, \dots, x_{ip})^T$ et le centre $(c_k = c_{k1}, \dots, c_{kp})^T$. Soit $M_{ik} \subset 1, \dots, p$ le sous-ensemble de coordonnées où x_i ou c_k ont des valeurs manquantes. La distance entre x_i et c_k est ajustée de la manière suivante :

$$S_{i,k} = \sum_{j \notin M_{ik}} w_j (x_{ij} - c_{kj})^2,$$

où $S_{i,k} = \sum_{j=1}^p w_j / \sum_{j \notin M_{ik}} w_j$.

Après avoir présenté en détail les différentes méthodes de regroupement, à savoir le k-means, le Trimmed K-means, le SK-means et le RSKC, il est nécessaire de faire une analyse comparative de ces approches. Dans le chapitre à venir, nous allons passer en revue chacune de ces méthodes dans le contexte de données simulées. Cette démarche nous permettra d'évaluer leur performance dans des conditions contrôlées et variées afin de mieux appréhender leurs comportements.

Chapitre 4

Applications des algorithmes aux données simulées

Dans ce chapitre, nous présentons les résultats d’une étude basée sur plusieurs simulations conçues pour analyser en profondeur les propriétés des algorithmes introduits dans le chapitre 2 3, tout en les comparant entre eux. Cette section est structurée comme suit :

Cadre de l’étude Avant d’entrer dans les détails des résultats, nous établissons le cadre de notre étude. Nous considérons deux cas distincts, à savoir :

Cas A : Nombre d’observations supérieur au nombre de variables ($n > p$)

Cas B : Nombre de variables supérieur au nombre d’observations ($n < p$)

Scénarios étudiés Dans chacun de ces cas, nous examinons trois scénarios différents pour évaluer la performance des algorithmes sous diverses conditions :

Scénario 1 : Données sans valeurs aberrantes

Scénario 2 : Présence de valeurs aberrantes parmi les variables pertinentes (Modèle 1).

Scénario 3 : Présence de valeurs aberrantes parmi les variables bruitées (modèle 2)

Notons que certains paramètres seront choisis et fixés par nous même dans le but de voir si ces algorithmes arrivent aux résultats escomptés.

Cette partie à été réalisée entièrement dans R Studio et le code se trouve en annexe.

Mesures d’évaluation Dans notre contexte où les véritables étiquettes de nos données sont connues, une mesure efficace pour évaluer la performance de nos algorithmes est le taux d’erreur de classification (CER), tel que présenté dans l’article de référence [12] définit comme suit : soient deux partitions P et Q, où P pourrait représenter la partition initiale et Q celle obtenue en appliquant un algorithme de partitionnement. Supposons que $1_{P(i,i')}$ soit un indicateur permettant de déterminer si la partition P regroupe les observations i et i' dans la même classe, et $1_{Q(i,i')}$ est défini de manière analogue pour la partition Q. Le CER est calculé comme $\sum_{i \neq i'} |1_{P(i,i')} - 1_{Q(i,i')}| / \binom{n}{2}$, où n est le nombre total d’observations.

Le CER varie entre 0 et 1. Une valeur de 0 signifie que les partitions P et Q correspondent parfaitement, tandis qu'une valeur élevée indique un désaccord entre les deux partitions.

4.1 Cas A : $n > p$

Les données simulées dans cette étude se composent d'un ensemble de $n = 1200$ observations, caractérisées par $p = 600$ variables générées à partir de distributions normales multi-variées, et avec une matrice de covariance égale à l'identité. Dans le cadre de la formation de l'échantillon, nous constituons trois classes de dimensions équivalentes. Cette formation repose sur la détermination du vecteur moyen μ , calculé comme étant le produit de μ par le vecteur b , où $b \in \mathbb{R}^{600}$ possède ses 50 premières composantes égales à l'unité, suivies de 550 zéros. Les valeurs spécifiques de μ sont attribuées selon les paramètres -2 , 0 et 2 , correspondant respectivement à chaque classe.

Il convient de noter que les distinctions entre ces classes sont entièrement définies par les cinquante premières variables, identifiées sous le terme de “variables de regroupement”, tandis que les 550 variables restantes se manifestent sous forme de bruit, caractérisées par $\mu = 0$ et $\sigma = 1$ pour leur distribution.

Pour une meilleure appréhension du phénomène, considérons initialement le scénario où seules les deux premières variables ont le pouvoir de caractériser les classes, tandis que les autres contribuent au bruit. Procédons à une comparaison graphique de nos méthodes. La figure 4.1.1 met en évidence, comme attendu, que les variables X_1 et X_2 sont capables de discriminer nettement les trois classes, que ce soit lorsqu'elles sont analysées individuellement (par projection sur l'une d'elles) ou en combinaison. Cependant, par exemple, les variables X_3 et X_{60} , supposées être du bruit, n'offrent pas la capacité de distinguer les différentes classes.

Cette observation illustre la prédominance discriminante des variables X_1 et X_2 par rapport aux autres variables, et souligne leur rôle crucial dans la caractérisation des classes.

Regardons maintenant ce qui se passe dans nos scénarios.

Sans valeurs aberrantes : Dans l'absence de valeurs aberrantes, la figure 4.1.2 affiche les nuages de points des deux premières variables (les plus discriminantes pour distinguer les classes) générés par les quatre algorithmes que nous soumettons à une comparaison avec les données originales. On observe que les algorithmes K-means et Trimmed K-means ne parviennent pas à séparer nettement les trois classes, tandis que les méthodes SK-means et RSKC semblent réussir à produire une partition satisfaisante en trois classes distinctes.

Avec valeurs aberrantes À présent, concentrons notre attention sur deux variables spécifiques, par exemple les variables X_1 et X_2 , qui sont considérées comme étant des variables de regroupement. Dans ce contexte, nous introduisons des valeurs aberrantes en remplaçant 10 observations (de 500 à 619, soit 0.833%) par des valeurs provenant d'une distribution uniforme dans l'intervalle $[-19, -10]$. À travers l'observation de la figure 4.1.3,

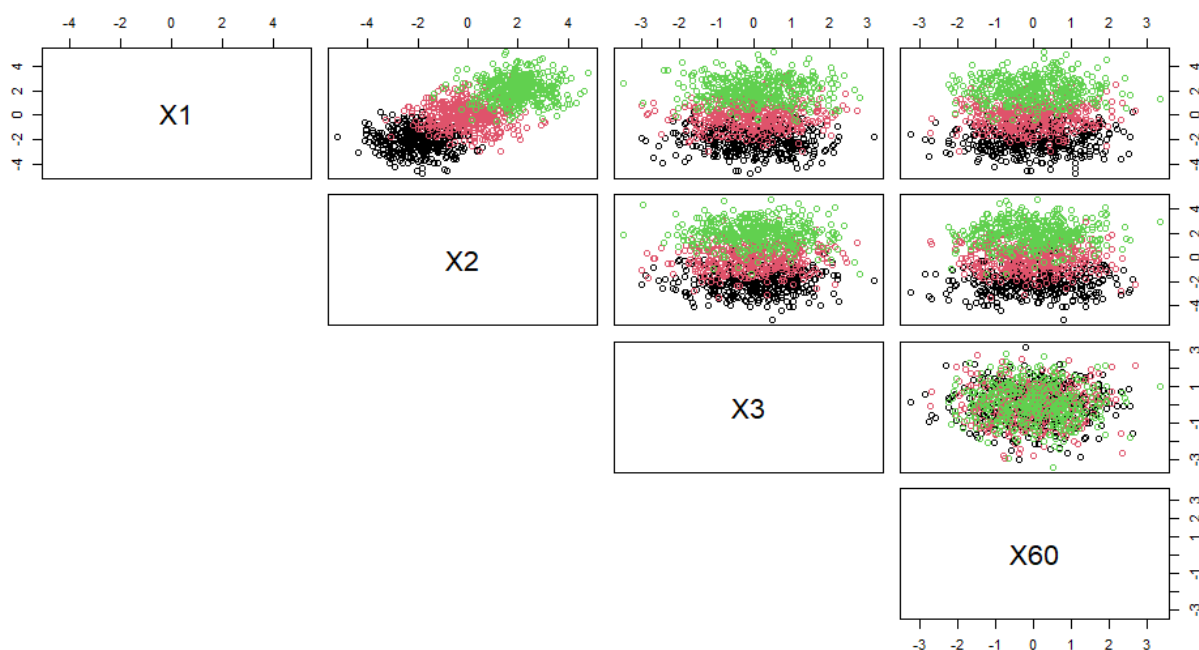


FIGURE 4.1.1 – Exploration de quelques variables discriminantes dans les données initiales.

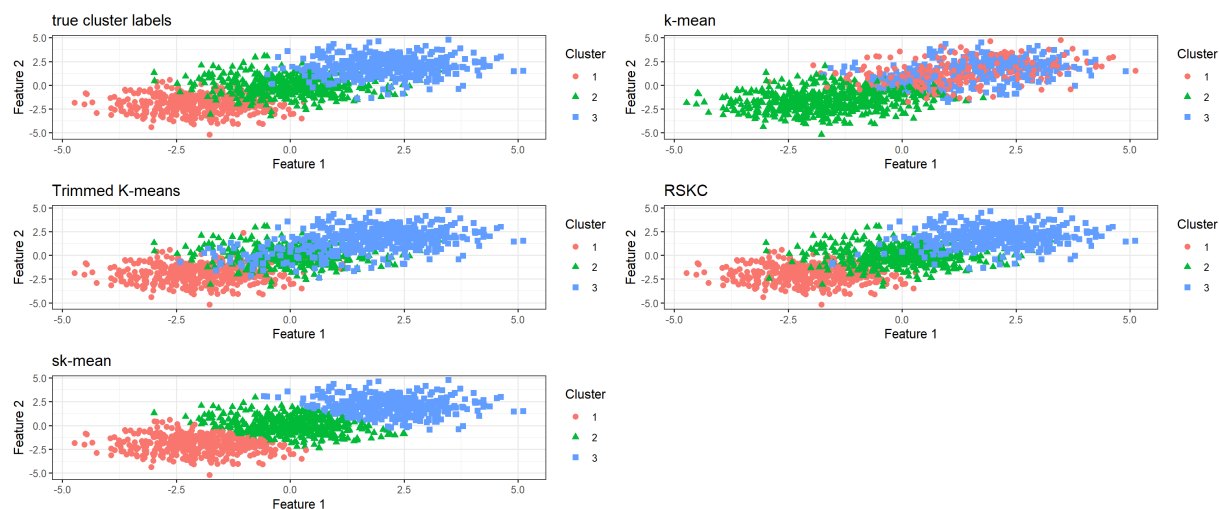


FIGURE 4.1.2 – Nuages de points permettant de comparer graphiquement les performances des différents algorithmes (sans valeurs aberrantes). La partition présente dans les données originales sert de référence.

nous pouvons discerner que les algorithmes K-means et trimmed K-means et SK-means échouent. En effet les algorithmes K-means et SK-means étant très sensibles aux valeurs aberrantes forment trois classes en considérant les 10 observations aberrantes comme une classe. En ce qui concerne l'algorithme Trimmed K-means, bien que les 3 classes soient

identifiées, leur distinction ne s'avère pas claire. En revanche, seul le RSKC réussit à distinguer les trois classes de manière adéquate en attribuant les valeurs aberrantes à la classe la plus proche. Ainsi, il apparaît que la version robuste du modèle s'avère performante en présence de valeurs aberrantes.

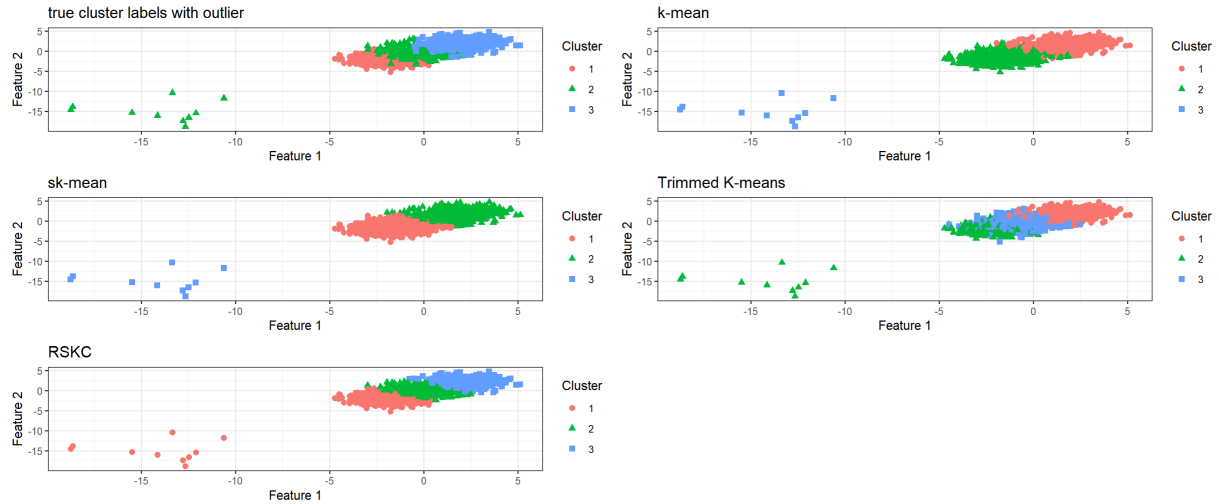


FIGURE 4.1.3 – Nuages de points permettant de comparer graphiquement les performances des différents algorithmes (avec valeurs aberrantes). La partition présente dans les données originales sert de référence.

4.1.1 Investigations des méthodes parcimonieuses et comparaisons avec plus de deux variables de regroupement

Dans cette section nous allons chercher les meilleures valeurs des paramètres s et α pour chaque méthode. Reconsidérons la simulation de départ mais cette fois ci avec 50 variables de regroupement sur les 600. Une approche simple pour sélectionner ces paramètres consiste à en tester plusieurs, à exécuter les algorithmes et à choisir les combinaisons qui produisent les meilleurs résultats, c'est-à-dire celles qui maximisent la fonction objectif. Une autre façon est présentée dans l'article [12] et propose d'utiliser la statistique Gap pour choisir le paramètre s . Nous l'avons présenté au chapitre précédent 3.3.3.

4.1.1.1 Sans valeurs aberrantes

1. La méthode SK-means

Nous souhaitons étudier l'impact du paramètre s sur nos résultats. Pour ce faire, nous effectuons une analyse en variant s sur plusieurs valeurs (de 3 à 9, avec une longueur de 15 dans notre contexte). Cette variation de s est réalisée en utilisant la statistique Gap, employée conjointement avec la fonction `KMeansSparseCluster` du package SPARCL de R Studio et avec une utilisation de 3 permutations.

La Figure 4.1.4 illustre la relation entre la taille de s et la proportion de poids non nuls. Lorsque s est de faible valeur (par exemple 1 ou 3), le nombre de poids non nuls est limité. À mesure que la valeur de s augmente, le nombre de poids non nuls augmente également. Cependant, il devient apparent qu'au-delà d'une certaine valeur de s , la variation dans le nombre de poids non nuls devient moins significative. Cette tendance est exemplifiée par le cas de $s = 8$, où l'ensemble des 50 premiers poids non nuls semblent converger autour de 0.15.

Dans la Figure 4.1.5 (à gauche), nous tracions l'évolution de la fonction Gap en fonction du nombre de variables de regroupement. L'analyse de cette courbe révèle une croissance rapide dans l'intervalle $[0, 50]$ suivie d'une croissance plus lente au-delà de 50 variables pertinentes. Cela renforce notre choix de sélectionner graphiquement 50 variables de regroupement.

Le graphique de droite dans cette même figure représente la distribution des poids. Il semble que les 50 premiers poids soient concentrés autour de 0.15 tandis que les autres poids semblent être proches de zéro. Cette observation nous permet d'affirmer que toute variable pertinente dispose d'un poids supérieur à 0.1. Cette valeur seuil sera d'une grande utilité pour déterminer numériquement le nombre de variables pertinentes sélectionnées par les approches parcimonieuses.

Enfin, dans ce contexte, le paramètre de s qui offre les meilleurs résultats s'établit à environ 7.286.

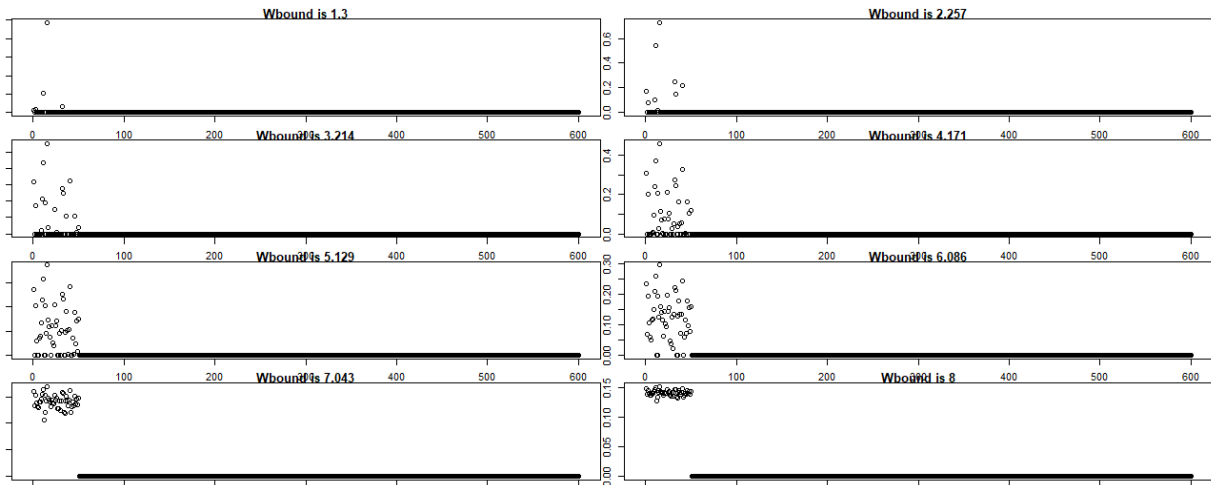
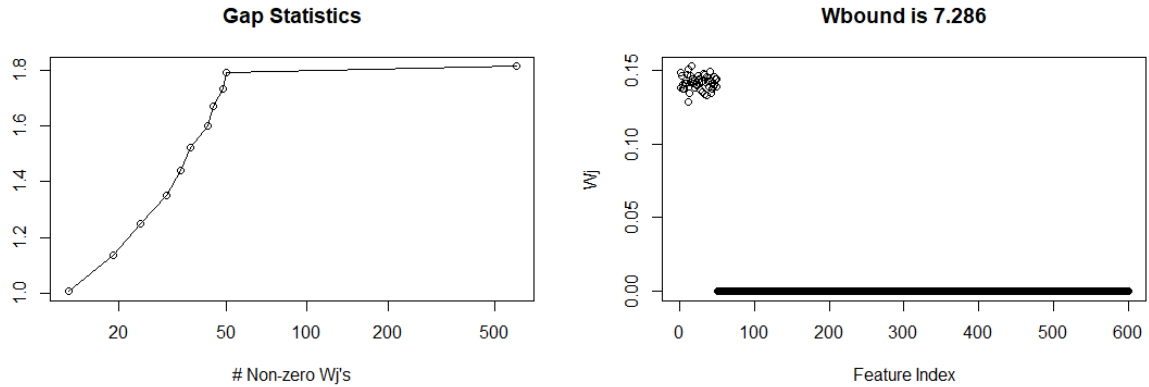


FIGURE 4.1.4 – Distribution des poids pour la valeur optimale de s

2. Les méthodes Trimmed K-means et RSK-means

En plus du paramètre s , la méthode RSK-means est également influencée par le paramètre α . Étant donné l'absence de méthode établie dans la littérature pour déterminer α , nous sommes contraints d'explorer plusieurs combinaisons de valeurs pour s et α afin de sélectionner les meilleures options. Le Tableau 4.1.1 présente le nombre de variables non nulles obtenues en expérimentant différentes valeurs de s



(a) Statistique de Gap en fonction du nombre de poids non nuls

(b) Distribution des poids

FIGURE 4.1.5 – Choix des paramètres du sparse k-mmean

et α . Les résultats indiquent qu'une augmentation de s de 6 à 8, indépendamment de la valeur de α , entraîne généralement une augmentation du nombre de variables pertinentes de 45 à 600. Ceci confirme que la plage optimale pour s se situe entre 6 et 8. De manière prévisible, le nombre de variables pertinentes augmente à mesure que s augmente.

Quant à la Tableau 4.1.2, elle présente les valeurs du taux d'erreur de classification (CER) obtenues pour ces mêmes combinaisons de valeurs. On observe qu'à partir de $s = 4$, quelle que soit la valeur de α , le CER est nul. Cette observation s'explique par l'absence de valeurs aberrantes dans notre contexte, et pourtant l'algorithme est contraint de supprimer une certaine proportion des valeurs aberrantes (déterminée par α).

Alpha	valeurs de s							$s = 16$
	$s = 2$	$s = 4$	$s = 6$	$s = 8$	$s = 10$	$s = 12$	$s = 14$	
$\alpha=0.05$	7	25	45	600	600	600	600	600
$\alpha=0.1$	6	24	46	600	600	600	600	600
$\alpha=0.2$	7	35	47	600	600	600	600	600

TABLE 4.1.1 – Table du nombre de variables non nuls obtenu en combinant les valeurs de s et α

3. Comparaison des différentes méthodes Dans cette section, nous procédons à une comparaison entre les algorithmes en fixant les paramètres suivants : $s = 7, 286$ pour le S-kmeans et RSKC, $\alpha = 0.05$ pour le Trimmed K-means et RSKC. La valeur de α choisie correspond à la plus petite valeur non nulle parmi celles que nous avons

Alpha	valeurs de s							s=16
	s=2	s=4	s=6	s=8	s=10	s=12	s=14	
$\alpha = 0.05$	0.0313	0	0	0	0	0	0	0
$\alpha = 0.1$	0.0312	0	0	0	0	0	0	0
$\alpha = 0.2$	0.0335	0	0	0	0	0	0	0

TABLE 4.1.2 – Table des valeurs du CER obtenues en combinant les valeurs de s et α

précédemment testées, compte tenu de l'absence de valeurs aberrantes dans notre scénario actuel.

La Tableau 4.1.3 présente les résultats de cette comparaison. La deuxième colonne affiche les valeurs moyennes du taux d'erreur de classification (CER), tandis que les écarts types correspondants sont entre parenthèses. La dernière colonne présente les moyennes du nombre de variables pertinentes. Ces résultats sont basés sur 100 simulations de notre base de données.

L'analyse de ces résultats permet de constater que le K-means standard produit des performances relativement moins bonnes, bien que généralement acceptables. En revanche, le SK-means et le RSKC parviennent à détecter approximativement le nombre attendu de variables pertinentes (environ 50), tout en maintenant des valeurs de CER raisonnables.

Sans valeurs aberr	CER (sd)	nbre de variables pertinentes (sd)
K-means	0.0558(0.1122)	600(0)
TK-means	0(0)	600(0)
SK-mean	0(0)	50.03(6.6613)
RSKC	1.11e-5(1.11e-4)	49.97(6.54812)

TABLE 4.1.3 – Table de comparaison des valeurs du CER et du nombre de variables pertinentes par méthode de regroupement. les valeurs dans les parenthèses représentent les écarts types

4.1.1.2 Avec valeurs aberrantes

Dans ce cas nous distinguerons deux situations :

modèle 1 : Valeurs aberrantes dans les variables pertinentes : X_4 et X_5 . Remplaçons 10% d'observations (120 observations) issues d'une distribution uniforme dans l'intervalle $[-19, -10]$.

Modèle 2 : Valeurs aberrantes dans les variables non pertinentes (bruits) : X_{60} et X_{61} . Remplaçons 10% d'observations (120 observations) issues d'une distribution uniforme dans l'intervalle $[-19, -10]$.

L'idée est de voir comment les valeurs aberrantes introduites dans chaque type de variable peuvent impacter ces algorithmes. Nous distinguerons toujours ces deux modèles à chaque fois que nous parlerons des valeurs aberrantes dans la suite. Il faut également noter que le paramètre α est connu et vaut 0.1 ; nous aurons plus besoin de le déterminer comme dans le cas précédent.

1. Sparse K-means

Modèle1 : La Figure 4.1.6 (à gauche) suggère que le nombre optimal de variables pertinentes pour atteindre le gap maximal se situe autour de 50. Par ailleurs, la partie droite de cette même figure montre que tous les poids non nuls sont regroupés autour de 0.1, et que l'ensemble des poids semblent être supérieurs à 0.05. Cette dernière valeur est adoptée comme seuil pour l'identification des variables pertinentes. la somme de tous ces poids vaut 4.4408.

En ce qui concerne le paramètre s , les résultats indiquent que le meilleur choix se situe à environ 4.714.

Modèle 2 : La Figure 4.1.7 (à gauche) suggère que le nombre optimal de variables pertinentes pour atteindre le gap maximal se situe autour de 52. Cette observation peut être justifiée par la présence d'entrées aberrantes affectant deux variables non-pertinentes. L'algorithme attribue des poids élevés à ces variables en raison de leurs variations interclasses, ce qui impacte le nombre optimal de variables pertinentes.

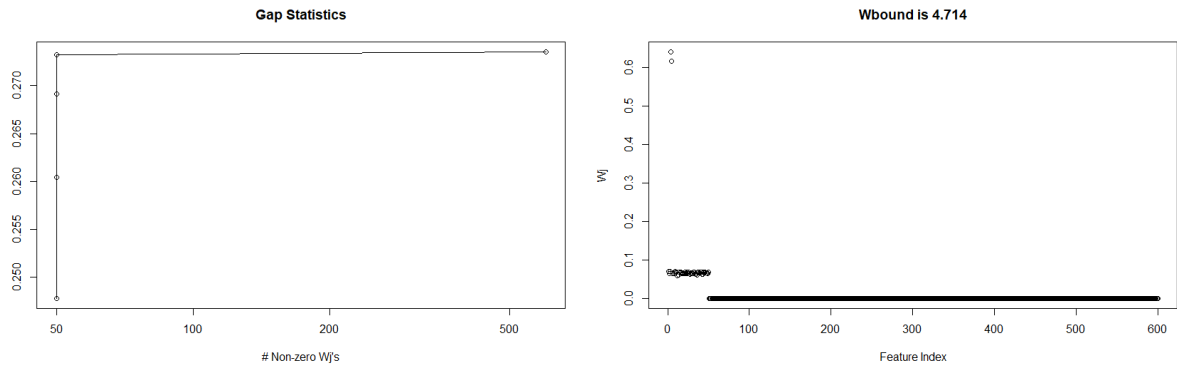
La partie droite de cette même figure révèle que tous les poids non nuls sont concentrés autour de 0.1, et que l'ensemble des poids semblent être supérieurs à 0.05. Cette valeur est retenue comme seuil pour l'identification des variables pertinentes. la somme de tous ces poids vaut 4.8303.

Concernant le paramètre s , les résultats suggèrent que le meilleur choix se situe à environ 5.143.

2. **Comparaison des différentes méthodes** Rappelons que les valeurs du paramètre s utilisées pour la comparaison dans cette section sont respectivement 4.714 pour le modèle 1 et 5.143 pour le modèle 2 et la valeur de α vaut 0.1.

La Tableau 4.1.4 présente les résultats de cette comparaison. La deuxième colonne affiche les valeurs moyennes du taux d'erreur de classification (CER), tandis que les écarts types correspondants sont entre parenthèses. La dernière colonne présente les moyennes du nombre de variables pertinentes. Ces résultats sont basés également sur 100 simulations de notre base de données.

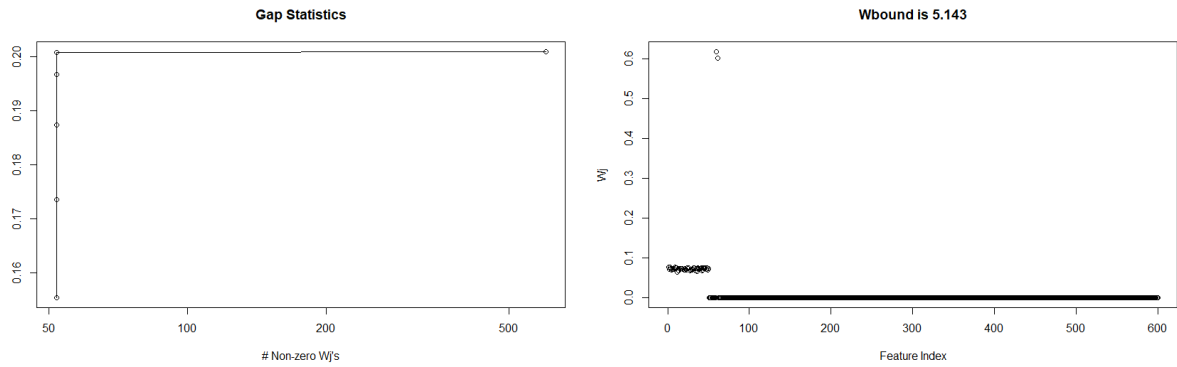
L'analyse de ces résultats permet de constater que : pour le modèle 1, le SK-means produit des performances relativement moins bonnes ; pour le modèle 2, le K-means standard s'en sort mieux que les méthodes Trimmed K-means et SK-means. En revanche, peu importe le modèle, l'algorithme RSKC donne des meilleurs résultats même si le nombre de variables pertinentes attendu a légèrement augmenté.



(a) Statistique de Gap en fonction du nombre de poids non nuls

(b) Distribution des poids

FIGURE 4.1.6 – Choix des paramètres du sparse k-mmean du modèle 1



(a) Statistique de Gap en fonction du nombre de poids non nuls

(b) Distribution des poids

FIGURE 4.1.7 – Choix des paramètres du sparse k-mmean du modèle 2

4.2 Cas B : $n < p$

Les données simulées dans ce cas sont composées de $n = 60$ observations et $p = 600$ variables, générées selon les mêmes paramètres que la simulation précédente. Les caractéristiques de ces données restent inchangées, avec les 50 premières variables jouant le rôle de variables de regroupement, tandis que les 550 variables restantes sont des bruits, suivant une distribution de moyenne $\mu = 0$ et d'écart-type $\sigma = 1$.

	Méthodes	CER (sd)	nbre de variables pertinentes (sd)
modèle 1	K-means	0,098(0,11)	600(0)
	TK-means	0,0208(0,0311)	600(0)
	SK-mean	0,2(0,0672)	47,44(9,1046)
	RSKC	0,0017(0,0054)	48(0)
modèle 2	K-means	0,106(0,114)	600(0)
	TK-means	0,185(0,0613)	600(0)
	SK-mean	1,1749(0,0729)	48,71(10,4120)
	RSKC	0,0005(0,0015)	55(0)

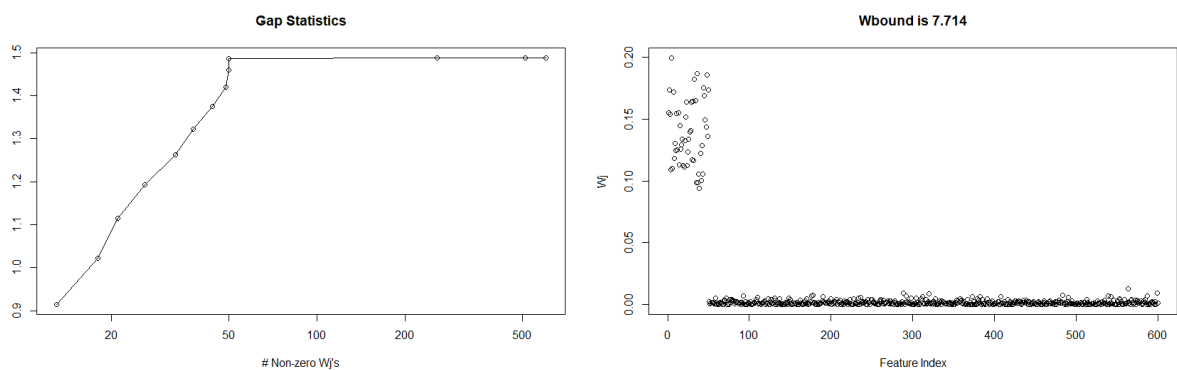
TABLE 4.1.4 – Table de comparaison des valeurs du CER et du nombre de variables pertinentes par méthode de regroupement. les valeurs dans les parenthèses représentent les écarts types

4.2.1 Investigations des méthodes parcimonieuses et comparaisons

4.2.1.1 Sans valeurs aberrantes

Choix du paramètre s Dans la Figure 4.2.1 (à gauche), on peut observer clairement qu’au-delà des 50 premières variables avec des poids non nuls, la statistique Gap présente une évolution pratiquement constante. Le Gap maximal atteint est de 1.4879. La partie droite de cette même figure présente la distribution des poids. On peut remarquer que les 50 premiers poids se situent autour de 0.15, tous étant supérieurs à 0.05. En revanche, les poids restants semblent nuls. Un seuil de 0.05 sera utilisé pour la sélection des variables pertinentes. la somme totale des poids vaut 7.7137.

En ce qui concerne le paramètre s , les résultats indiquent que le choix optimal se situe autour de 7,714.



(a) Statistique de Gap en fonction du nombre de poids non nuls

(b) Distribution des poids

FIGURE 4.2.1 – Choix des paramètres du sparse k-means ($n < p$) sans valeurs aberrantes

Comparaison Une comparaison entre les différents algorithmes a été réalisée en fixant les paramètres suivants : $s = 7.7142$ pour le S-kmeans et RSKC, $\alpha = 0.05$ pour le Trimmed K-means et RSKC. Le choix de ces paramètres repose sur des considérations similaires à celles du cas A - Scénario 1.

Les résultats du tableau 4.2.1 montrent que, dans l'ensemble, tous les algorithmes produisent de bons résultats. Si nous examinons les performances avec une grande précision, le K-means semble avoir légèrement plus d'erreurs, avec un taux d'erreur de classification (CER) de 0,04. Le SK-means et le RSKC, en revanche, parviennent à identifier de manière satisfaisante les différentes classes. Ces deux derniers algorithmes présentent l'avantage d'identifier un nombre de variables pertinentes proche de 50.

Sans valeurs aberr	CER (sd)	nbre de variables pertinentes (sd)
K-means	0,0398(0,0985)	600(0)
TK-means	0,00312(0,0186)	600(0)
SK-mean	0(0)	51,16(5,967)
RSKC	0(0)	50,81(6,9277)

TABLE 4.2.1 – Table de comparaison des valeurs du CER moyen et du nombre moyen de variables pertinentes par méthode de regroupement ($n < p$ sans valeurs aberrantes). les valeurs dans les parenthèses représentent les écarts types. Résultats obtenus après 100 simulations de nos données

4.2.1.2 Avec valeurs aberrantes

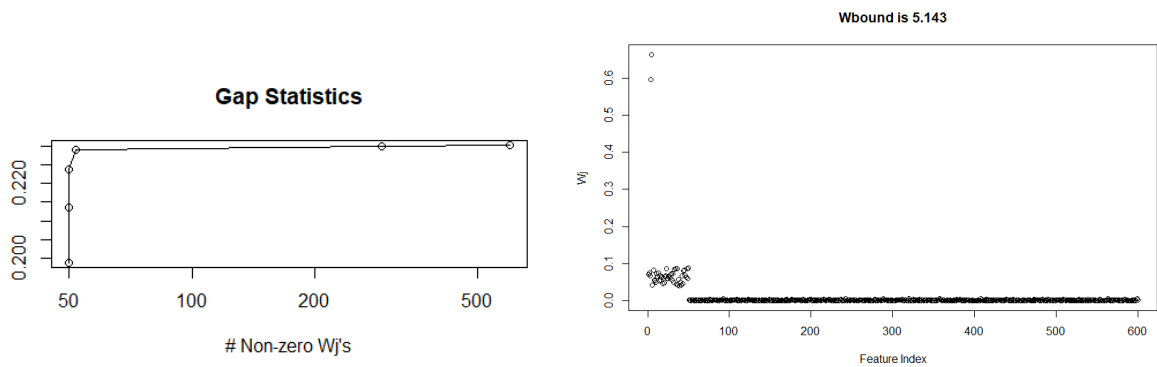
Nous conservons toujours nos deux modèles de valeurs aberrantes.

1. Choix du paramètre s

Modèle 1 : Dans la Figure 4.2.2 (à gauche), on peut observer qu'au-delà d'un certain nombre de variables pertinentes (environ 53), la statistique Gap ne présente qu'une évolution presque constante, avec une valeur maximale de 0.2434. Ceci indique une surestimation du nombre de variables pertinentes. Seulement trois variables supplémentaires ont été ajoutées pour la distinction des classes. La partie droite de cette même figure présente la distribution des poids. On constate que tous les poids non nuls sont regroupés près de 0.1, et que l'ensemble des poids semblent être supérieurs à 0.05. Cette valeur est retenue comme seuil pour l'identification des variables pertinentes. la somme totale des poids vaut 4.8179.

Quant au paramètre s , les résultats indiquent que le meilleur choix se situe autour de 5.143. Cette valeur sera utilisée pour la comparaison entre les algorithmes.

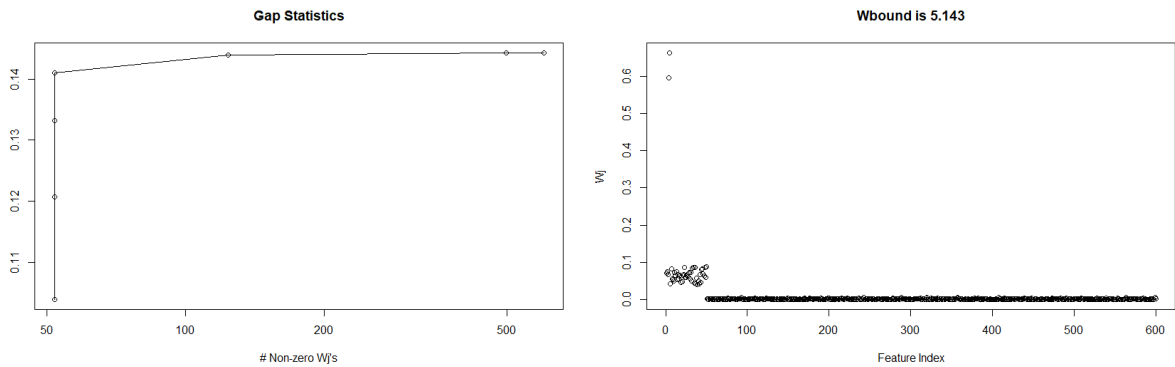
Modèle 2 : Dans la figure 4.2.3 à gauche, on peut voir que 52 variables pertinentes permettent d'avoir une statistique de gap de 0.1551 ; cette valeur voulue légèrement jusqu'à 0.1580 pour un nombre de variables de 143. Ce nombre est quand même surprenant car dans le modèle 2, les valeurs aberrantes ne concernent que deux variables non pertinentes donc on s'attend à avoir deux variables pertinentes supplémentaires. On voit également à droite de cette même figure que tous les poids non-nuls sont distribués près de 0.1 et que tous les poids semblent être au dessus de 0.05. Cette valeur sera retenue comme seuil de détermination des variables pertinentes. Le meilleur paramètre de s dans ce cas est 5.571 et sera utilisé pour la comparaison des algorithmes.



(a) Statistique de Gap en fonction du nombre de poids non nuls

(b) Distribution des poids

FIGURE 4.2.2 – Choix des paramètres du Sk-mmean du modèle 1



(a) Statistique de Gap en fonction du nombre de poids non nuls

(b) Distribution des poids

FIGURE 4.2.3 – Choix des paramètres du Sk-mmean du modèle 2

2. Comparaison

Le Tableau 4.2.2 met en évidence que, peu importe le modèle d'introduction des valeurs aberrantes, le RSKC affiche les meilleurs résultats en termes de taux d'erreur de classification (CER) comparé aux autres algorithmes. En effet, pour le modèle 1, le RSKC obtient un CER de 0,0469 et pour le modèle 2 un CER de 0,0018. Le Trimmed K-means suit de près avec des performances également solides ($CER = 0,0428$ pour le modèle 1 et $CER = 0,0154$ pour le modèle 2). Le K-means, quant à lui, présente des résultats moins satisfaisants comparativement aux trois autres méthodes, même si son taux d'erreur n'est pas excessivement élevé.

	Méthodes	CER (sd)	nbre de variables pertinentes (sd)
modèle 1	K-means	0,114(0,119)	600(0)
	TK-means	0,0428(0,0665)	600(0)
	SK-mean	0,149(0,112)	44,77(10,751)
	RSKC	0,0469(0,0538)	51,81(89,9629)
modèle 2	K-means	0,11(0,119)	600(0)
	TK-means	0,0154(0,0531)	600(0)
	SK-mean	0,124(0,108)	48,9(7,41)
	RSKC	0,00184(0,013)	49,56(2,85)

TABLE 4.2.2 – Table de comparaison des valeurs du CER moyen et du nombre moyen de variables pertinentes par méthode de regroupement ($n < p$ avec valeurs aberrantes). les valeurs dans les parenthèses représentent les écarts types. Résultats obtenus après 100 simulations de nos données

Nous pouvons conclure que dans l'ensemble des situations, l'algorithme RSKC présente les performances les plus élevées en comparaison avec les autres méthodes. Lorsque $n > p$ en l'absence de valeurs aberrantes, les performances de ces algorithmes sont comparables. Cependant, en présence de valeurs aberrantes, le Trimmed K-means (TK-means) peut être une alternative viable au K-means standard, car le K-means et le SK-means semblent être très affectés par la présence de valeurs aberrantes. Lorsque $n < p$ sans valeurs aberrantes, le Trimmed K-means et le SK-means présentent des performances comparables, tandis qu'en présence de valeurs aberrantes, il est préférable d'utiliser le K-means plutôt que le SK-means. En revanche, le RSKC présente des meilleurs résultats dans tous les cas et tous les scénarios.

Nous notons également que, en l'absence de valeurs aberrantes, le RSKC et le SK-means attribuent approximativement le même nombre de variables pertinentes. Cependant, en présence de valeurs aberrantes, le RSKC semble être moins perturbé dans la sélection des variables pertinentes, tandis que le SK-means détecte moins de variables pertinentes.

Chapitre 5

Application des différentes approches aux données textuelles

Après avoir effectué une comparaison des différentes méthodes sur des données que nous avons générées, avec une structure préétablie et des étiquettes bien connues pour faciliter la comparaison, nous allons maintenant appliquer ces approches à des données réelles. Dans ce contexte, nous ne disposerons pas initialement d’informations préalables sur la structure des données, et nous allons délibérément masquer certaines informations pour évaluer les performances des différentes méthodes. L’objectif est de déterminer quelles approches produisent les meilleurs résultats et de vérifier si les conclusions que nous avons tirées dans le chapitre précédent restent valables dans le contexte des données textuelles.

5.1 Présentation de la base de données

Description des la base de données La base de données utilisée pour l’évaluation de nos algorithmes est la célèbre base de données “20 Newsgroups”. Cette base de données est largement utilisée dans les domaines de l’apprentissage automatique et du traitement du langage naturel. Elle a été créée en 1995 par Ken Lang et Peter Turney et comprend des documents textuels provenant de 20 catégories distinctes, couvrant un large éventail de sujets tels que la politique, les sports, la technologie, la religion, les sciences, etc. Chaque catégorie contient plusieurs documents textuels associés qui sont généralement des articles de presse, des messages de forums Usenet, des courriels et autres formes de textes courts. La seule langue utilisée est l’anglais. Cette base de données a été initialement conçue pour des tâches de classification de texte et est librement accessible en ligne depuis diverses sources. Dans notre cas, nous avons accédé à la base de données directement depuis la bibliothèque Python scikit-learn.

En raison de contraintes de mémoire, nous avons choisi de travailler avec seulement trois catégories (religion, politique de guerre et le matériel informatique IBM) afin de réduire le nombre de documents à traiter. Cette sélection nous a donné un total de 1735 documents. La fonction `fetch20newsgroups` de scikit-learn, qui fournit cette base de données, inclut

également une variable appelée “target”. Nous avons transformé cette base de données en un “DataFrame” en ajoutant cette variable, comme illustré dans la Figure 5.1.1. Bien que la variable “target” sera ignorée dans les algorithmes de regroupement que nous allons utiliser, elle se révélera très utile lors de l’évaluation de nos différents algorithmes.

	text	target
0	From: rayssd!esther@uunet.uu.net (Esther A. Pa...	2
1	From: jayne@mmalt.guild.org (Jayne Kulikauskas...	2
2	From: fcrary@ucsu.Colorado.EDU (Frank Crary)\n...	3
3	From: reedr@cgsvox.claremont.edu\nSubject: Re:...	2
4	From: eggertj@moses.atc.ll.mit.edu (Jim Eggert...	2

FIGURE 5.1.1 – Présentation des 5 premières lignes de la base de données.

Statistiques descriptives Le graphique à barres présenté dans la Figure 5.1.2 révèle que les catégories 1 et 2 sont presque également représentées avec un nombre de documents juste en dessous de 600, tandis que la catégorie 3 est moins représentée mais demeure proche des autres avec environ 550 documents. Nous pouvons anticiper que cette distribution sera maintenue par nos algorithmes si ces derniers sont efficaces dans le contexte des données textuelles.

La Figure 5.1.3 illustre la distribution du nombre de caractères (ou longueur) dans chaque document, en utilisant le logarithme de cette longueur pour réduire l’échelle. Nous observons que les 1735 articles possèdent un corpus de texte avec une longueur qui suit une distribution normale, centrée autour d’une moyenne de 315 mots et avec un écart-type de 404 mots.

5.2 Pré-traitement des données

L’étape de prétraitement des données revêt une importance cruciale dans la classification ou le regroupement des données textuelles, car elle peut avoir un impact significatif sur les résultats obtenus. Dans cette phase (voir [30]), il s’agit de définir la structure que nos données doivent prendre. Cela comprend principalement la transformation du texte en tokens (un concept qui sera défini dans la première partie de cette section), l’élimination des divers types de bruit ou filtrage, la suppression des mots vides et la normalisation.

Il est essentiel de comprendre la notion de **document** : il représente une séquence de caractères (par exemple : une ligne de texte, un paragraphe etc). La Figure 5.2.1 présente les deux premiers paragraphes du premier document avant tout nettoyage. Nous avons souligné quelques éléments non pertinents pour notre analyse, qui doivent être écartés. Dans

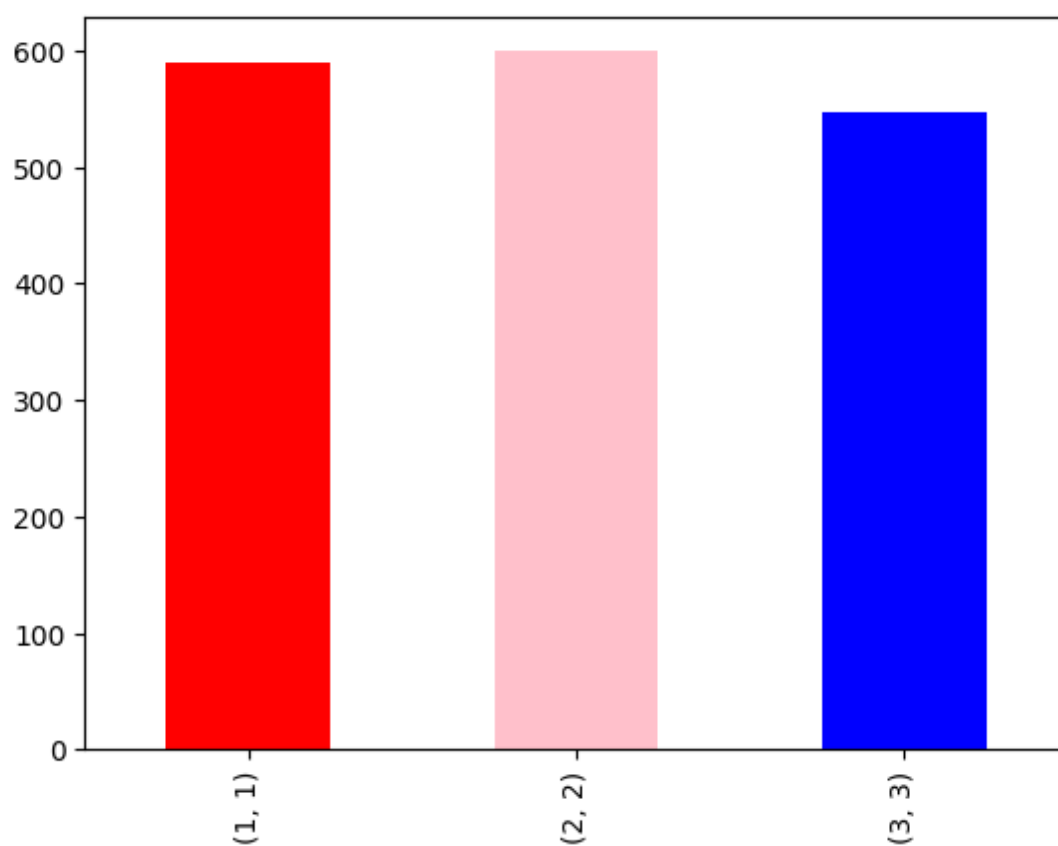


FIGURE 5.1.2 – Visualisation du nombre de documents par catégorie.

les sections suivantes, nous allons appliquer chacune de ces étapes de prétraitement afin d'obtenir une structure exploitable pour nos données.

1. **Segmentation de mots.** L'une des premières étapes couramment utilisées en prétraitement consiste à diviser chaque document en une liste d'occurrences appelée unité lexicale (token). Généralement, chaque unité lexicale correspond à un mot, souvent séparé par des espaces dans un texte. Cependant, il est important de noter que cette séparation simple ne suffit pas. Par conséquent, il est crucial d'analyser attentivement la structure des phrases.

Par exemple, si nous examinons la première ligne du texte présenté dans la Figure 5.2.1 : **From : rayssd !esther@uunet.uu.net (Esther A. Paris)**. Une segmentation à partir des espaces donnerait les mots : **"From :"** , **"rayssd !esther@uunet.uu.net"** , **"(Ether"** , **"A."** et **"Paris)"**. Cela souligne la nécessité de traiter également les ponctuations et autres caractères (comme les parenthèses, les apostrophes et dans certains cas les chiffres) en tant qu'unités lexicales, tout en tenant compte des exceptions. Par exemple, le mot **"google.com"** ne devrait pas être divisé en **"google"** et **"com"**. De plus, il peut être souhaitable de conserver les points faisant partie des abréviations.

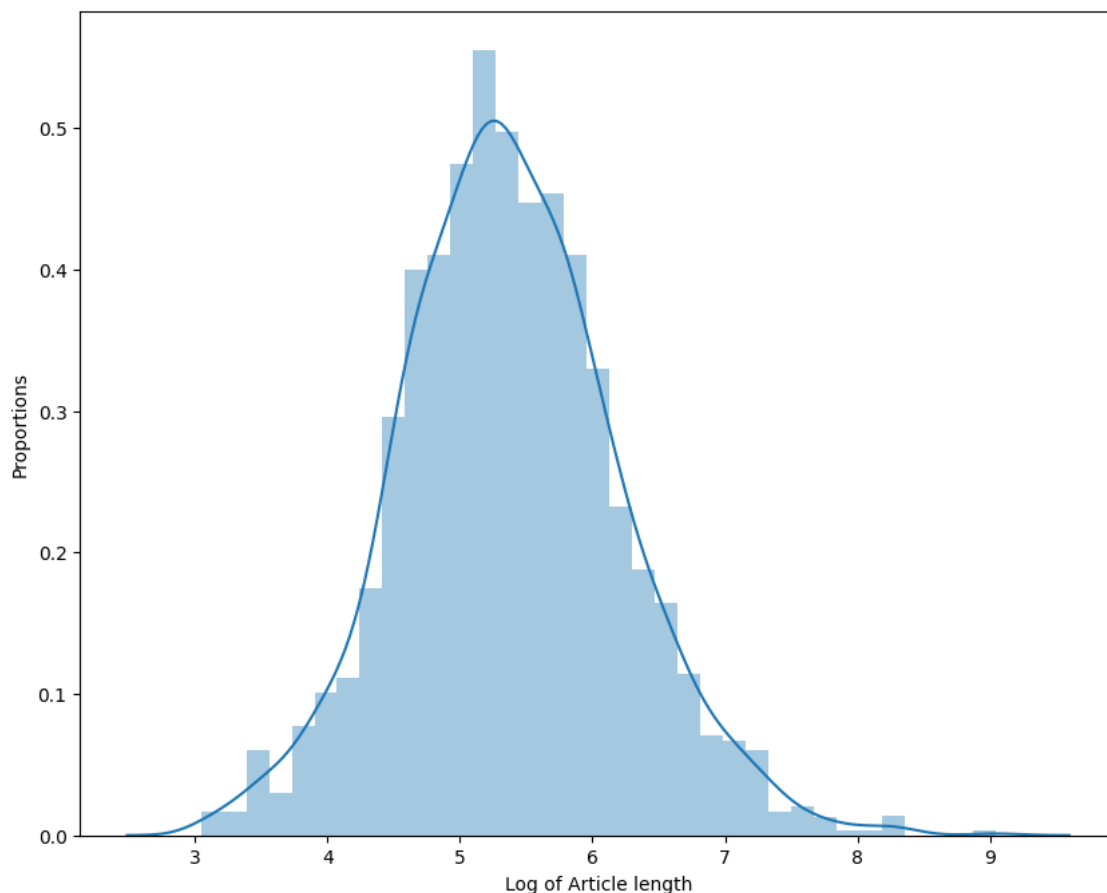


FIGURE 5.1.3 – Distribution de la longueur des documents : le logarithme du nombre de caractères de chaque documents ou article.

```
From: rayssd!esther@uunet.uu.net (Esther A. Paris)
Subject: harrassed at work, could use some prayers
Reply-To: esther@demand.ed.ray.com
Organization: Raytheon Equipment Division, Marlboro, MA
Lines: 110
```

```
My news feed is broken and I haven't received any new news in 243 hours
(more than 10 days). So, if you reply to this, please send private
email to the address esther@demand.ed.ray.com -- I have set the
Reply-To line to have that address but I don't know if it will work.
```

FIGURE 5.2.1 – Présentation des deux premiers paragraphes du premier document avant nettoyage.

Lors de la phase de segmentation, il peut également être nécessaire de convertir les

clitiques¹ en leurs formes originales. Par exemple, **I'd** pourrait être transformé en **I would**. De même, il peut être utile de traiter certains tokens comme des collocations. Par exemple, les mots "**Naturel language processing**" devraient être conservés ensemble pour préserver leur signification et faire référence au traitement du langage naturel.

Le choix des règles de segmentation dépend de l'application et de la langue. Dans le contexte du regroupement de documents, une analyse sémantique approfondie des phrases peut ne pas être nécessaire. Nous nous concentrons sur la conservation des mots pertinents pour différencier nos documents.

2. **La normalisation** Dans certaines applications, les variations morphologiques d'un mot ne sont pas utiles, voir même nuisent. Exemple : En recherche d'information, les accords ne sont pas importants ; si je cherche le mot **numbers**, je souhaiterais que tous les documents contenant le mot **number** soient retournés. On aimerait normaliser les formes prises d'un mot pour éviter les variations parasites. La normalisation peut se faire de 2 manières :

- La Racinisation (stemming) : Il s'agit de retirer les affixes d'un mot pour obtenir une forme la plus proche du radical ou racine. Souvent on se limite simplement au retrait de son affixe. Exemple : [consultant, consulting, consultants] – > consul. Il faut noter que le résultat obtenu n'est pas nécessairement un mot.

- La lemmatisation : Il s'agit d'obtenir l'entrée du dictionnaire d'un mot. Exemple : **Jumping** ->**Jump**, **numbers**->**number** ; **over** reste inchangé. Il faut noter qu'après la lemmatisation, on doit obtenir un mot.

3. **Le filtrage** Le processus de nettoyage implique principalement la suppression de tout ce qui n'est pas identifié comme un mot, en anglais dans notre cas. Dans le texte présenté dans la 5.2.1, cela inclut les expressions soulignées en bleu, ainsi que les ponctuations et autres caractères spéciaux. Cependant, le nettoyage va au-delà de cela. Dans notre situation, nous souhaitons supprimer les nombres ainsi que les mots vides (stop words). Les mots vides sont des mots qui apparaissent fréquemment dans l'ensemble des documents et n'apportent pas de contribution significative à la classification des documents. Ils incluent des termes tels que "a", "an", "and", "the", "on", "of", "for", "from", "but", "not", "as", etc.

Après avoir exécuté toutes ces étapes de prétraitement, le texte aboutit à une forme similaire à celle illustrée dans la figure 5.2.2. On peut constater que le texte a été considérablement réduit et ne contient que des mots jugés pertinents. Cependant, à ce stade, nos algorithmes de regroupement ne peuvent pas encore être appliqués, car ils requièrent des données numériques en entrée. Dans la section suivante, nous allons convertir ces données textuelles en données numériques, plus précisément en une matrice de nombres réels.

1. les clitiques sont des éléments linguistiques qui ont besoin d'être liés à d'autres mots pour exprimer leur sens complet et qui sont souvent moins autonomes que les mots pleins.

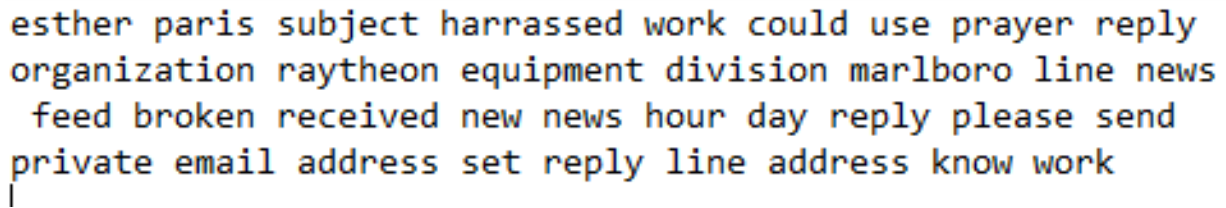


FIGURE 5.2.2 – Présentation des deux premiers paragraphes du premier document après nettoyage.

5.3 Représentation vectorielle des données

Dans la section précédente (voir le livre [31] pour plus de détails), nous avons procédé à la segmentation de notre corpus d'entraînement en mots. À présent, nous allons définir un vocabulaire en collectant tous les mots distincts présents dans les textes obtenus après prétraitement. Concrètement, cela revient à former une liste de tous les mots uniques présents dans ce corpus², et en même temps, à remplacer chaque mot peu ou trop présent dans le corpus d'entraînement par `<unk>` ou `<oov>` (out of vocabulary); exemple : `[and,good,...,<unk>]` est une représentation du vocabulaire. Cette approche permet de gérer efficacement les mots inconnus dans le corpus d'entraînement, tout en préservant la cohérence et la capacité de généralisation des modèles³ (voir le livre [31], page 48).

Soit M le nombre de documents, et d_i représentant le document i . L'objectif est de représenter chaque document d_i ($w_1, w_2, \dots, w_n$) par un vecteur de dimension n (où n est la taille du vocabulaire). Ce vecteur est défini comme suit : $d_i = (x_1, x_2, \dots, x_n)$, où $1 \leq j \leq n$ et x_j est le poids associé au mot w_j dans le document i . L'objectif ultime est de créer une matrice de dimensions $M \times n$ appelée matrice terme-document.

5.3.1 Modèle de Sac des mots (Bag of words)

Pour débiter, clarifions le concept de **modèle de langue (ML)** : Il s'agit d'une approche qui attribue une probabilité à une séquence de mots dans un langage, basée sur une estimation de maximum de vraisemblance. Le modèle de langue apprend à prédire la probabilité d'une séquence s à n mots par :

$$P(s) = P(w_1, w_2, \dots, w_n).$$

Cette probabilité peut être exprimée comme un produit de probabilités conditionnelles.

2. le corpus est une collection de documents dans un langage donné.

3. NB : afin d'avoir une taille raisonnable, le vocabulaire est fixé par nous. On décide alors de remplacer les mots peu fréquents (min_{freq}) ou trop fréquents (max_{freq}) par `<unk>`. Cela permet au vocabulaire de s'ouvrir à tout nouveau corpus.

Définition : Étant donné deux évènements A et B , la probabilité de A sachant B avec $p(B) > 0$ est définie par :

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

En utilisant cette définition, on a :

$$\begin{aligned} P(s) &= P(w_1, w_2, \dots, w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \dots P(w_n|w_1w_2 \dots w_{n-1}) \\ &= \prod_i P(w_i|w_1w_2 \dots w_{i-1}) \end{aligned}$$

Ce modèle dépend de plusieurs paramètres. Cela suppose pour le mot w_i , il faut considérer les $i - 1$ mots qui le précèdent considéré comme **contexte ou historique**. Ainsi donc, pour un modèle N -gram, l'historique représente les $N - 1$ mots précédents. On a l'approximation suivante :

$$P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-N+1} \dots w_{i-1})$$

Cette expression est connue sous le nom de postulat de Markov (Markov assumption). Finalement on a :

$$\begin{aligned} P(s) &= P(w_1, w_2, \dots, w_n) \\ &\approx \prod_i P(w_i|w_{i-N+1} \dots w_{i-1}) \end{aligned}$$

Basée sur une estimation de maximum de vraisemblance, on a :

$$P(w_i|w_{i-N+1} \dots, w_{i-1}) = \frac{C(w_{i-N+1}, \dots, w_{i-1}, w_i)}{C(w_{i-N+1}, \dots, w_{i-1})},$$

où $C(w_{i-N+1}, \dots, w_{i-1}, w_i)$ est le nombre de fois que l'historique $w_{i-N+1}, \dots, w_{i-1}$ est suivie du mot w_i et $C(w_{i-N+1}, \dots, w_{i-1}) = \sum_i C(w_{i-N+1}, \dots, w_{i-1}, w_i)$ avec $w_i \in V$. Pour $N = 1$, nous avons un modèle unigram ou 1-gram défini comme suit :

$$\begin{aligned} P(s) &\approx P(w_1)P(w_2)P(w_3) \dots P(w_n) \\ &\approx \prod_i^n P(w_i) \end{aligned}$$

La représentation en sac de mots repose sur ce modèle. C'est une approche simple pour convertir un document en une matrice de termes. L'idée sous-jacente est que la signification d'un document est encapsulée uniquement dans les termes qu'il contient, et que des documents similaires contiennent des termes similaires. En conséquence, un document est représenté comme un ensemble non ordonné de mots, avec leurs fréquences respectives, tout en ignorant leur position. L'information sur l'ordre des mots est perdue, mais dans

notre contexte, cela n'a pas d'importance, car la structure grammaticale ne joue pas de rôle crucial.

Chaque mot w_j dans le vocabulaire est traité comme une variable avec sa propre fréquence x_j . Par conséquent, un document est représenté par un vecteur construit à partir des fréquences de ces mots (c'est-à-dire, nous comptons combien de fois chaque mot apparaît dans le document). La dimension de ce vecteur correspond à la taille du vocabulaire, qui dans notre cas est de 5590.

Par exemple, supposons qu'on a 3 commentaires de films :

doc 1 : *The plot of the movie was engaging and kept me hooked till the end.*

doc 2 : *The acting in the movie was impressive, especially the lead actor's performance.*

doc 3 : *I found the cinematography to be visually stunning and added to the overall experience.*

Nous n'appliquons aucun traitement, car c'est juste une illustration.

Bag of Words : { "The" : 8, "plot" : 1, "of" : 1, "movie" : 2, "was" : 2, "engaging" : 1, "and" : 2, "kept" : 1, "me" : 1, "hooked" : 1, "till" : 1, "end" : 1, "acting" : 1, "in" : 1, "film" : 1, "impressive" : 1, "especially" : 1, "lead" : 1, "actor's" : 1, "performance" : 1, "I" : 1, "found" : 1, "cinematography" : 1, "to" : 1, "be" : 1, "visually" : 1, "stunning" : 1, "added" : 1, "overall" : 1, "experience" : 1 }

L'ensemble des mots présents dans ce dictionnaire constitue le vocabulaire. La représentation matricielle de ce sac de mots est donnée par le tableau 5.3.1.

	the	plot	of	movie	was	engaging	and	...
Doc1	3	1	1	1	1	1	1	...
Doc2	3	0	0	1	0	0	0	...
Doc3	2	0	0	0	0	0	1	...

TABLE 5.3.1 – Représentation matricielle du sac des mots. 7 mots du vocabulaire ont été représentés.

Cependant, cette approche présente un inconvénient majeur : elle ne tient pas compte du contexte. Par exemple, si nous avons la séquence "New York", les mots "New" et "York" sont comptés individuellement. Pour surmonter ce problème, on peut envisager d'utiliser des modèles sacs de mots bigrammes, trigrammes, etc.

5.3.1.1 La pondération TF-IDF

Le modèle de sac de mots présente une limitation : les mots fonctionnels, également appelés mots-outils (function words), risquent d'avoir peu d'impact. Ces mots sont utilisés fréquemment dans presque tous les documents, ce qui signifie qu'ils ne sont pas particulièrement discriminants en ce qui concerne la nature ou la sémantique du sujet traité dans les documents. En revanche, les mots qui ont plus de potentiel sont ceux qui sont spécifiquement utilisés dans certains types de documents. Ainsi, l'objectif est de développer un modèle statistique qui puisse capturer ce comportement.

Pour cela, il faut introduire des poids pour chaque mot, et ces poids ajustés seront incorporés dans la représentation en sac de mots. Jusqu'à présent, le poids x_{ij} du j^e mot dans le vocabulaire est simplement sa fréquence dans le i^e document. On introduit le concept de m_j , qui représente le nombre de documents contenant au moins une occurrence du j^e mot dans le vocabulaire. Afin d'obtenir une valeur qui n'est pas dépendante du nombre total de documents, on utilise la proportion m_j/M , où M est le nombre total de documents, pour influencer la pondération. L'idée est que plus la proportion m_j/M est élevée, moins le poids $x_{i,j}$ de ce mot devrait être élevé dans la représentation en sac de mots.

La pondération TF-IDF est souvent divisée en deux parties :

TF (term frequency) : c'est le terme noté : $tf_{i,j}$ qui représente la fréquence du j^e mot du vocabulaire dans le i^e document défini par sa forme logarithme de la manière suivante :

$$tf_{i,j} = \log(\text{fréquence du mot} + 1).$$

Exemple : en appliquant cette expression dans la matrice 5.3.1, on obtient la matrice représentée dans le tableau 5.3.2. On peut remarquer que le mot "the" apparaît dans tous les documents. Il semble que ce mot ne soit pas discriminant.

	the	plot	of	movie	was	engaging	and	...
Doc1	0.60	0.30	0.30	0.30	0.30	0.30	0.30	...
Doc2	0.60	0	0	0.30	0	0	0	...
Doc3	0.48	0	0	0	0	0	0.30	...

TABLE 5.3.2 – Représentation matricielle du TF. 7 mots du vocabulaire ont été représentés.

IDF (inverse document frequency) : c'est le logarithme de l'inverse de la proportion m_j/M :

$$idf_j = \log\left(\frac{M}{m_j}\right)$$

Le tableau 5.3.3 le mot "the" a un poids nul.

	the	plot	of	movie	was	engaging	and	...
m_j	3	1	1	2	1	1	1	...
idf_j	0.0	0.48	0.48	0.18	0.48	0.48	0.48	...

TABLE 5.3.3 – Représentation matricielle du IDF. 7 mots du vocabulaire ont été représentés.

TF-IDF : c'est tout simplement le produit de $tf_{i,j}$ et de idf_j :

$$x_{ij} = tf_{i,j} \times idf_j$$

Pour finir notre illustration, on a donc la matrice présentée dans le tableau 5.3.4. On voit bien que contrairement aux résultats obtenus dans la représentation sac de mots, on a pu annuler l'impact du mot “the” qui ne permet pas de distinguer les documents.

	the	plot	of	movie	was	engaging	and	...
Doc1	0.0	0.14	0.14	0.05	0.14	0.14	0.14	...
Doc2	0.0	0.0	0.0	0.05	0.0	0.0	0.0	...
Doc3	0.0	0.0	0.0	0.0	0.0	0.0	0.14	...

TABLE 5.3.4 – Représentation matricielle du TF-IDF. 7 mots du vocabulaire ont été représentés.

La matrice résultante après avoir appliqué la pondération TF-IDF dans notre étude de cas est illustrée dans la figure 5.3.1. Nous y présentons les 5 premières lignes de cette matrice, mettant en avant les cinq mots les plus significatifs du premier document. Nous remarquons que cette matrice est très creuse ; elle contient peu de valeurs non nulles. Soit une densité de **1,7 %**.

	0	1	2	3	4	5	6	7	8	9	...	1725	1726	1727	1728	1729	1730	1731	1732	1733	1734
pray	0.248565	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
lunch	0.213136	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
sex	0.204454	0.0	0.0	0.0	0.0	0.000000	0.000000	0.050051	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
please	0.188577	0.0	0.0	0.0	0.0	0.036501	0.091673	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.085944	0.0	0.0	0.0	0.0
left	0.172084	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0

5 rows x 1735 columns

FIGURE 5.3.1 – Présentation des 5 premières lignes obtenues après pondération TF-IDF. Rappelons que la matrice entière a une dimension de (1735,5590)

5.4 Applications des différents algorithmes de regroupements

Dans cette section, nous procéderons à l'application de nos divers algorithmes de regroupement dans le but de déterminer les groupes de documents similaires. À travers l'utilisation du t-SNE 2.2.2 pour projeter nos documents sur un espace à deux dimensions, comme illustré dans la figure 5.4.1, nous pouvons déjà observer une séparation en trois classes de documents qui se distinguent verticalement. Nous aspirons à corroborer ces premiers résultats grâce à l'application de nos méthodes de regroupement.

5.4.1 Algorithme de K-means

Dans cette étape, nous allons d'abord appliquer l'algorithme K-means standard pour déterminer les étiquettes qui permettront de différencier les diverses classes de nos documents.

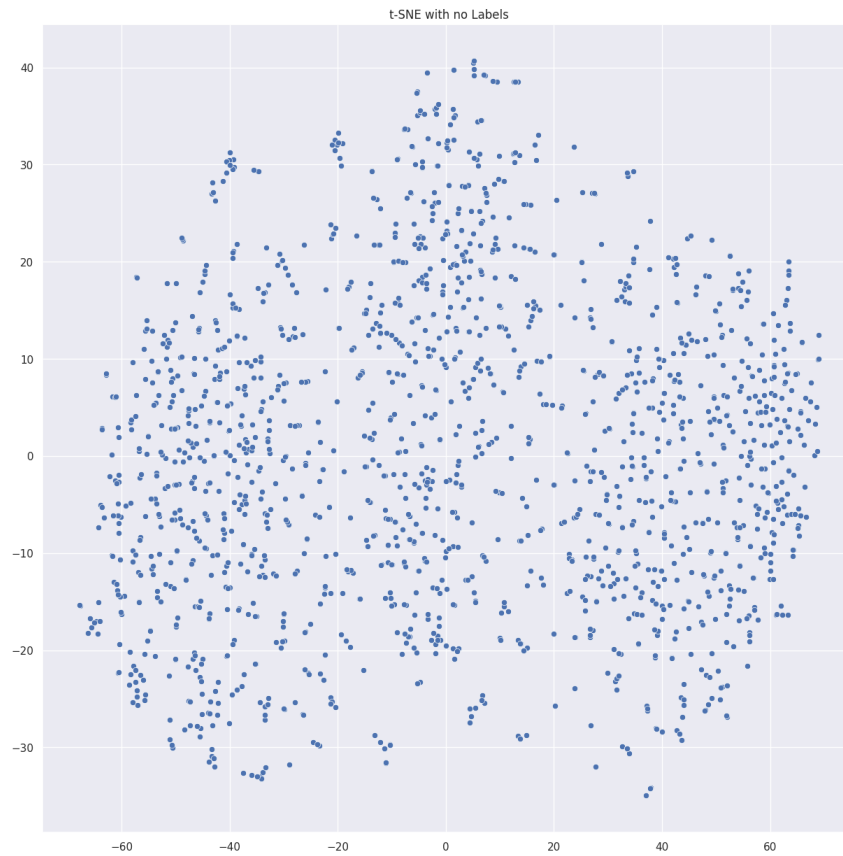


FIGURE 5.4.1 – t-sne sans labels.

Dans ce contexte spécifique, nous n'avons pas besoin de déterminer le nombre de classes car il est déjà connu à l'avance (3 classes). Par conséquent, nous fixons $K = 3$ et utilisons la distance euclidienne comme mesure de similarité pour tous les quatre algorithmes.

La visualisation graphique dans la figure 5.4.2, avec des annotations pour une meilleure compréhension, nous permet de mieux appréhender comment les articles sont regroupés. On remarque que l'algorithme K-means fournit les étiquettes qui nous permettent de distinguer clairement les différentes classes.

Cependant, il est important de noter que dans certains cas, certaines étiquettes colorées se retrouvent également dans d'autres groupes, bien que ces occurrences soient peu fréquentes. Cette observation peut être due aux variations introduites par le t-SNE et le K-means lorsqu'ils travaillent avec des dimensions supérieures. Les sujets abordés dans les articles ont souvent des thèmes qui se chevauchent, ce qui complique la séparation nette de ces groupes.

En se référant à la visualisation graphique de la figure 5.4.2, nous pouvons conclure que le K-means donne des résultats satisfaisantes, avec un taux d'erreur de classification (CER) de 0.034.



FIGURE 5.4.2 – T-sne avec les labels obtenus à partir du K-means.

5.4.2 Algorithme de Trimmed K-means

Pour l'algorithme de Trimmed K-means, la recherche de la proportion α optimale a été effectuée en faisant varier les valeurs suivantes : 0.05, 0.1, 0.15, 0.2. Les résultats des taux d'erreur de classification (CER) 4 pour chaque valeur de α sont résumés dans le tableau 5.4.1.

Alpha	0.05	0.1	0.15	0.2
CER	0.0439	0.031	0.0287	0.0294

TABLE 5.4.1 – TK-means : CER en fonction de la proportion α

En examinant les valeurs du CER dans ce tableau, on observe que toutes ces différentes valeurs de α peuvent être utilisées sans conséquence majeure dans nos résultats ; mais la valeur la plus élevée vaut 0.0287 et est obtenu pour $\alpha = 0.15$. On serait tenté de dire que cette valeur du CER selon nos résultats, diminue lorsqu'on augmente la valeur de α , mais on constate une légère augmentation lorsqu'on passe de 0.15 à 0.2.

Il apparaît donc qu'en considérant la méthode de TK-means, la meilleure valeur de α est

0.15. Cette valeur a permis d'obtenir la figure 5.4.3. Tout comme le K-means standard, cette approche permet de distinguer clairement nos différentes classes, même si quelques points se retrouvent dans d'autres classes. Ceci peut s'expliquer par les mêmes raisons exposées dans la section précédente.

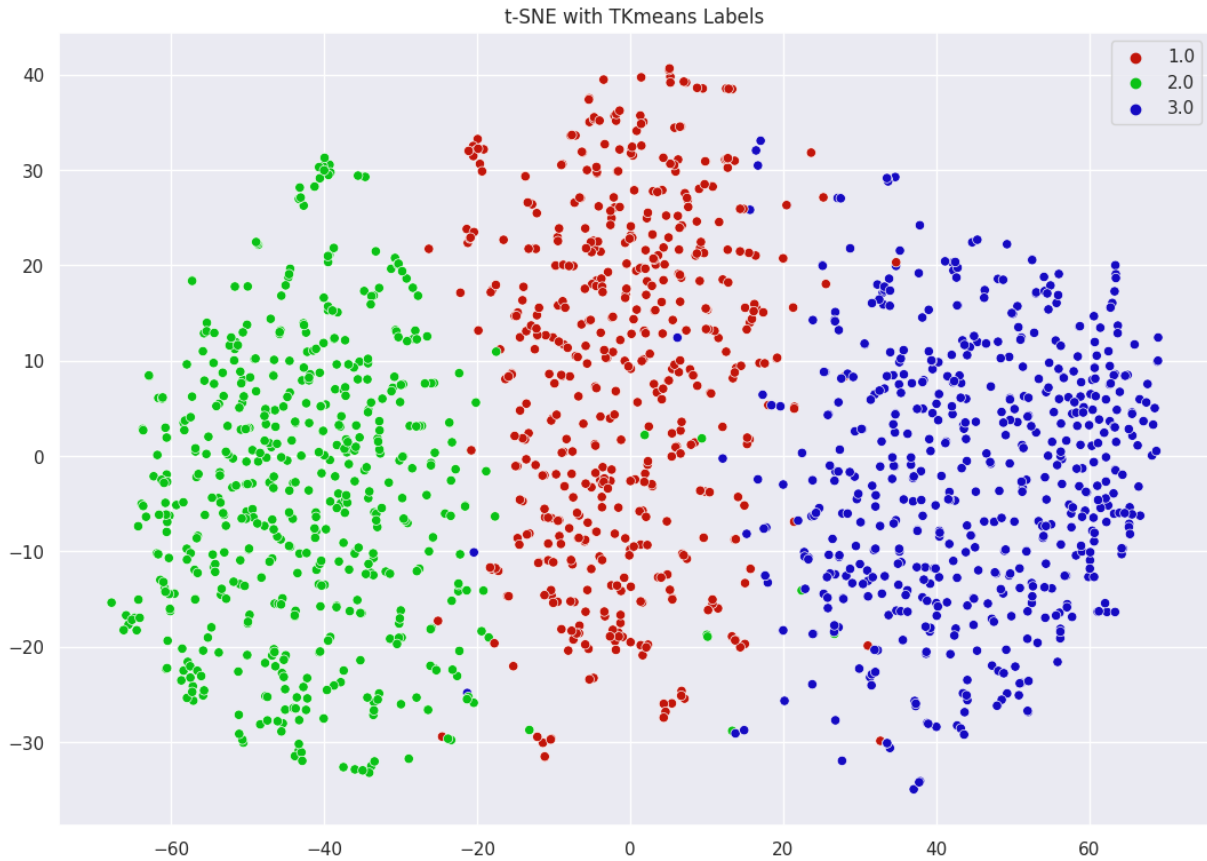


FIGURE 5.4.3 – t-sne avec les labels obtenus à partir du TK-means..

5.4.3 Algorithme de SK-means

Pour l'algorithme Sparse K-means (SK-means), le paramètre de parcimonie s a été testé en utilisant la statistique de Gap avec 15 valeurs allant de 3 à 9. La meilleure valeur obtenue pour s est 3. La distribution des poids pour cette approche est illustrée dans la figure 5.4.4. En examinant cette figure, on peut noter que à l'exception de deux poids qui sont supérieurs à 0.6, les autres poids semblent être proches de zéro. La somme de tous ces poids est de 2.6947. Cela indique que seulement deux caractéristiques sont jugées pertinentes pour distinguer les classes selon cette méthode, tandis que les autres sont considérées comme du bruit.

En utilisant la valeur optimale de s obtenue ($s = 3$), la projection des documents avec l'algorithme Sparse K-means a abouti à la figure 5.4.5. Cependant, cette projection

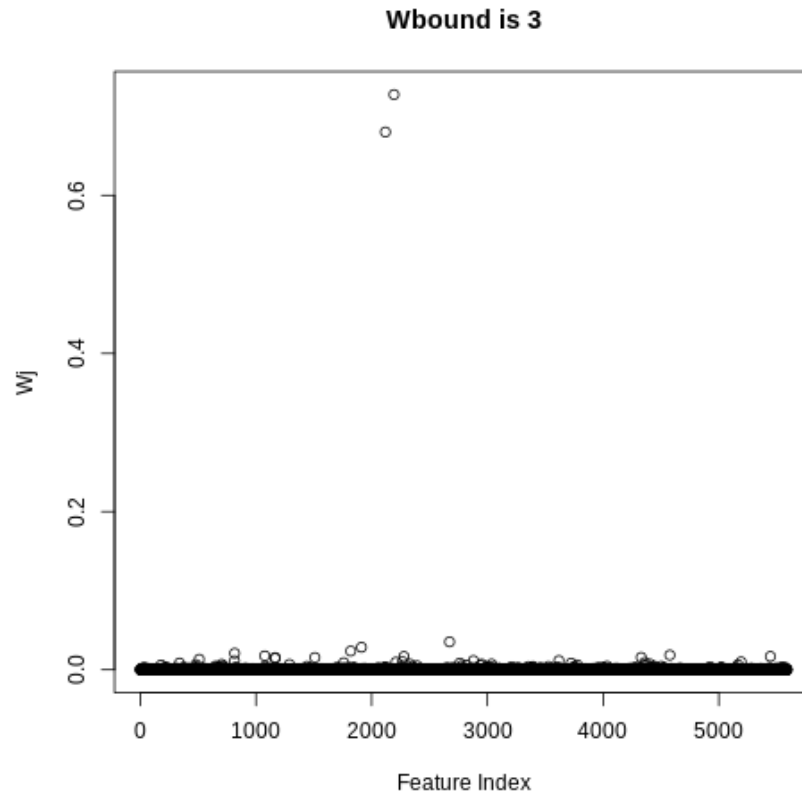


FIGURE 5.4.4 – Distribution des poids pour la valeur optimale de s dans le cas des données textuelles

ne semble pas offrir une bonne séparation entre les classes. On peut observer un fort chevauchement entre les différentes classes, ce qui suggère que cet algorithme a du mal à distinguer efficacement les classes. De plus, il y a une prédominance excessive dans l'une des classes (représentée en rouge), ce qui ne correspond pas aux proportions attendues d'après le graphe 5.1.2.

5.4.4 Algorithme de RSK-means

Dans cette approche, nous avons utilisé les valeurs optimales des paramètres obtenues dans les approches précédentes, à savoir $\alpha = 0.15$ et $s = 3$. La projection des documents obtenue avec l'algorithme RSK-means est présentée dans la figure 5.4.6. De manière similaire à l'approche SK-means, cette projection ne semble pas fournir une séparation claire entre les classes. On observe encore un chevauchement significatif entre les différentes classes, et une prédominance excessive dans l'une des classes (représentée en bleu), ce qui ne correspond pas aux proportions attendues.

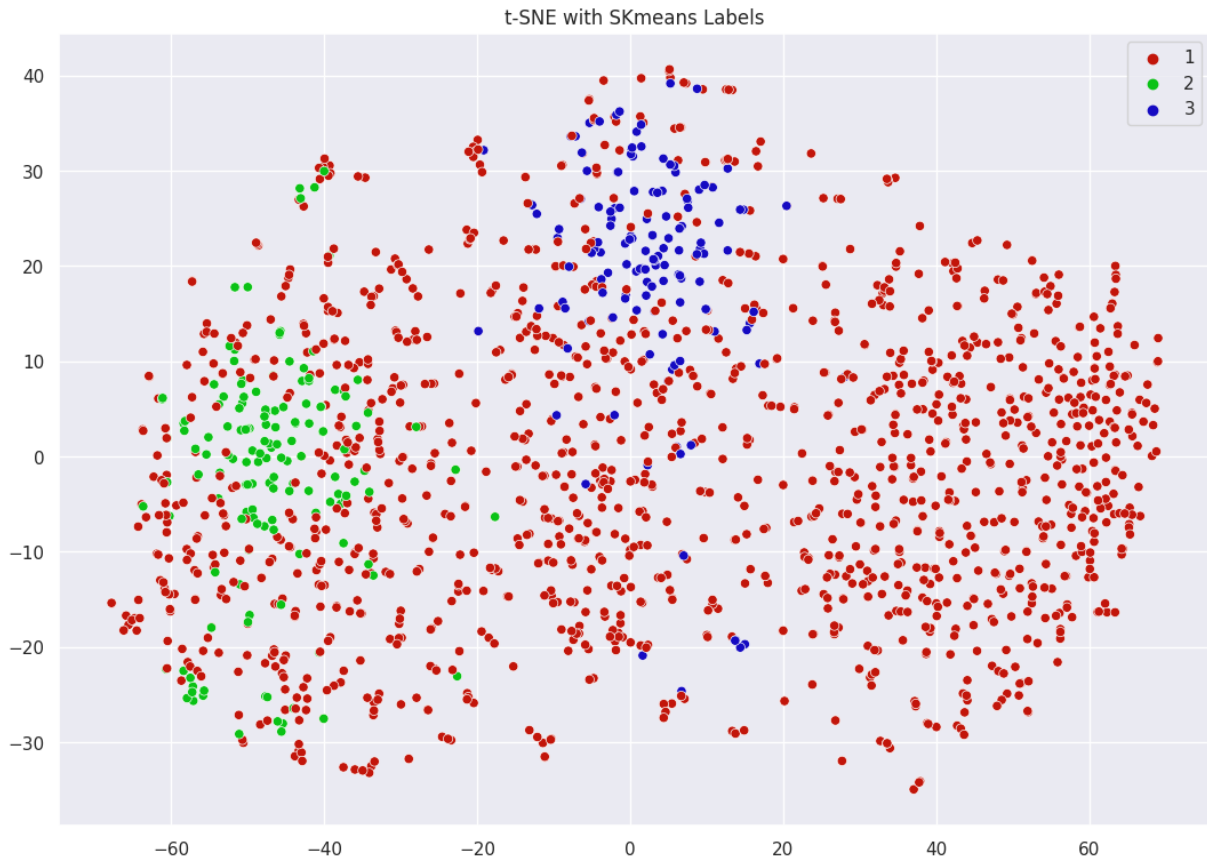


FIGURE 5.4.5 – T-sne avec les labels obtenus à partir du SK-means.

5.5 Comparaison des valeurs du CER issues des quatre méthodes et sélection de la meilleure méthode

Après avoir identifié précédemment la meilleure partition pour chaque approche, nous allons maintenant les comparer en fonction de leurs valeurs de CER et choisir celle qui présente le taux d'erreur le plus faible. Les résultats obtenus ont été résumés dans le tableau 5.5.1.

Sans surprise, en se basant sur les résultats précédents obtenus dans ce chapitre, nous constatons que les méthodes parcimonieuses (SK-means et RSK-means) ont les taux d'erreur les plus élevés, soit respectivement 0.5341 et 0.4651. Si l'on compare ces deux méthodes parcimonieuses, il est clair que la méthode RSK-means donne de meilleurs résultats. En revanche, les approches du K-means standard et du TK-means offrent de meilleurs résultats, comme nous l'avons observé à travers les projections T-SNE précédentes. Dans notre cas particulier, nous choisissons d'utiliser la méthode Trimmed K-means car elle présente le taux d'erreur le plus bas parmi les méthodes testées.

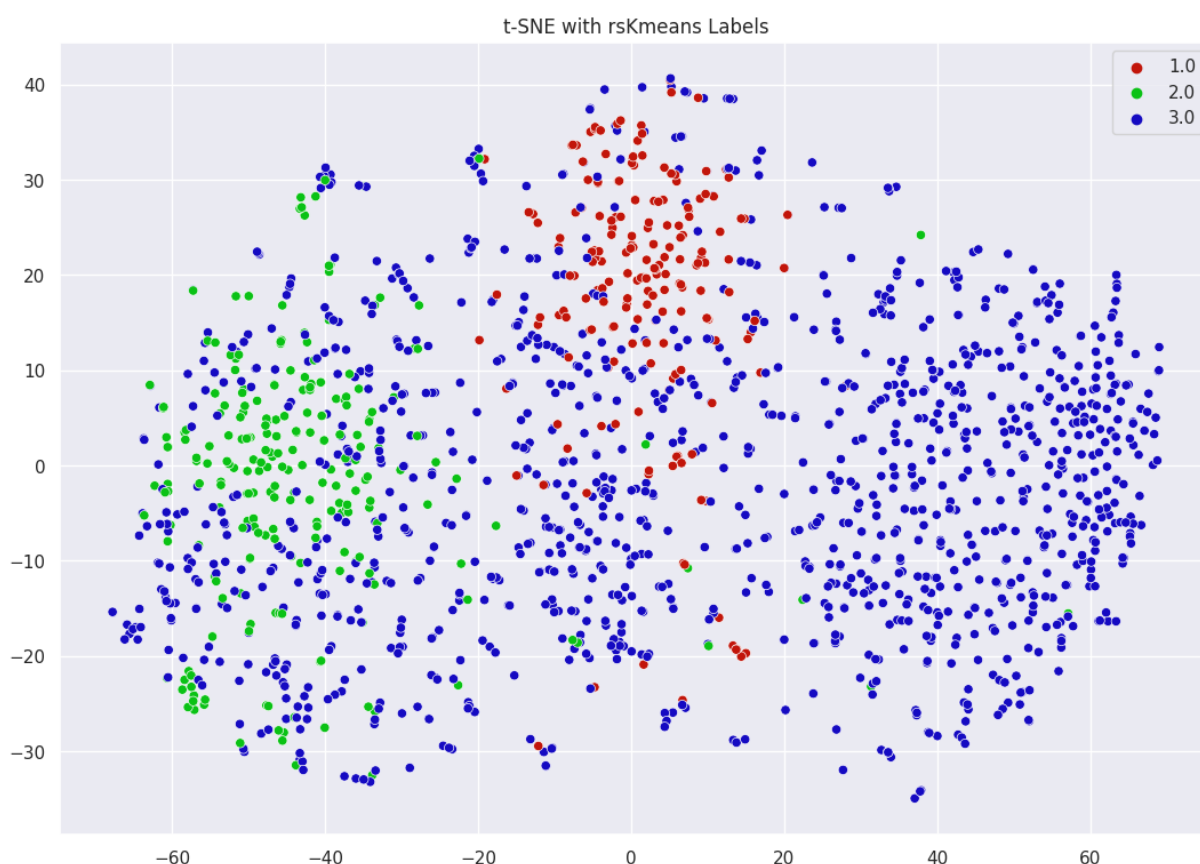


FIGURE 5.4.6 – T-sne avec les labels obtenus à partir du RSK-means.

	K-means	TK-means	SK-means	RSK-means
CER	0.034	0.0287	0.5341	0.4651

TABLE 5.5.1 – Comparaison des différentes méthodes en fonction du CER.

5.6 Analyse des classes obtenues

Après avoir identifié les différentes classes à l'aide de la méthode Trimmed K-means, nous allons maintenant nous pencher sur la caractérisation de ces classes. L'objectif est d'essayer d'identifier les différents sujets abordés par chacune de ces classes. Pour ce faire, nous allons examiner les mots les plus fréquents dans chaque classe.

Nous avons choisi de présenter graphiquement les 20 premiers mots de chaque classe en fonction de leur score moyen, qui représente la fréquence moyenne d'apparition de chaque mot dans la classe. Nous pouvons consulter la figure 5.6.1 pour visualiser ces informations. Cette analyse des mots les plus utilisés dans chaque classe peut nous donner des indications sur les thèmes dominants ou les sujets abordés par les documents au sein de chaque groupe.



FIGURE 5.6.1 – Représentation des mots communs les plus fréquent de chaque classe.

1. Pour la classe 0 : le mots le plus utilisé est le mot **god**, suivi des mots christian, jesus, church. On remarque aussi les mots comme : bible, christ, believe, etc. Ces termes évoquent clairement des sujets liés aux croyances religieuses et à la spiritualité.
2. Pour la classe 1 : le mot le plus utilisé est **gun** ; On remarque également les mots comme : weapon, fbi, firearm, law, government, fire, control, etc. Ces mots semblent pointer vers des thèmes associés à la politique de contrôle des armes à feu et à la sécurité nationale.
3. Pour la classe 2 : le mot le plus utilisé est la mot **drive** suivi des mots card, scsi, ide bus, system, controller, disk, monito, etc. Ces mots semblent être associés au domaine du matériel informatique et des périphériques.

Nous avons également présenté ces classes sous forme des nuages de mots. Les nuages de mots sont une excellente façon de visualiser et de communiquer les thèmes saillants dans chaque classe de documents comme le montre la figure 5.6.2. Dans cette représentation, la taille d'un mot indique sa fréquence ou son importance.

Dans les trois nuages de mots présents dans cette figure, nous pouvons voir que le premier nuage de mots affiche des mots-clés liés à la politique de contrôle des armes à feu et à la sécurité nationale., tandis que le deuxième nuage de mots affiche des mots liés à la religion et le dernier affiche les mots liés au domaine du matériel informatique et des périphériques. Ainsi, nos performances de regroupement semblent bonnes dans ces nuages de mots.

En somme, à travers l'exploitation d'une base de données textuelle regroupant un



FIGURE 5.6.2 – Nuage de mots les plus fréquents de chaque classe.

ensemble de documents classés en trois catégories, nous avons dans un premier temps ignoré les étiquettes. Puis nous avons entrepris un processus exhaustif de prétraitement comprenant la segmentation en mots, le filtrage, ainsi que l'élimination des mots vides. Ensuite, nous avons opté pour l'utilisation du TF-IDF qui nous a permis d'obtenir une représentation vectorielle adéquate. En poursuivant nos analyses, nous avons soumis ces données à nos quatre algorithmes distincts. Les résultats obtenus ont révélé que l'approche la plus performante était celle du Trimmed K-means. À partir des conclusions tirées de ce modèle, nous avons pu identifier trois sujets prédominants au sein des classes caractérisées : les croyances religieuses et la spiritualité, la politique de contrôle des armes à feu et la sécurité nationale et le domaine du matériel informatique et des périphériques. Ces découvertes témoignent de l'efficacité de la méthode employée, car nous avons dans une certaine mesure retrouvé les catégories choisies initialement.

Chapitre 6

Discussion/Conclusion

En conclusion, cette recherche avait pour objectif de pallier les limitations du K-means standard en contexte de haute dimension, en explorant des méthodes parcimonieuses capables d'identifier le sous-ensemble de variables responsables de la distinction entre les classes. Les différentes étapes entreprises dans cette thèse ont permis d'atteindre cet objectif de manière approfondie.

Dans un premier temps, les problématiques liées à l'analyse de données en haute dimension ont été exposées ; notamment la malédiction de la dimension. Les méthodes antérieures de regroupement ont été présentées à l'exemple de la réduction préalable de la dimension avant le regroupement comme première approche en situation de haute dimensionnalité.

Le K-means a été examiné dans sa forme traditionnelle en mettant en évidence ses avantages et ses limites en haute dimension. Partant du constat qu'il existe des configurations où seulement certaines caractéristiques ou variables permettent de distinguer les classes (variables de regroupement), la version robuste du K-means a été présentée (trimmed K-means) et les méthodes parcimonieuses telles que le SK-means et le RSKC ont été introduites comme des alternatives potentielles pour détecter ces variables de regroupement. Une analyse détaillée de ces méthodes a été menée pour comprendre leurs fonctionnements et leurs avantages.

En appliquant ces méthodes aux données simulées dans diverses configurations de comparaison et grâce au taux d'erreur de classification (CER), il a été déterminé que la méthode RSKC a émergé comme la plus performante ; démontrant sa capacité à identifier les variables pertinentes grâce à la régularisation de LASSO imposée.

Cependant, dans le contexte des données textuelles, dont nous ne connaissons pas la structure interne (contrairement aux données simulées) et qui en général ont une particularité d'avoir une faible densité (1.7 % dans notre cas), la méthode trimmed K-means s'est révélée être la plus adaptée contre toute attente, illustrant son aptitude à identifier les informations pertinentes dans ce contexte spécifique. Ce résultat peut se justifier par le fait que le K-means et sa version robuste Trimmed K-means ne prennent pas en compte les poids des variables et attribuent les mêmes poids à toutes les caractéristiques c'est-à-dire la même importance. De ce fait nous pensons que si en effet dans la base de données toutes les variables ont les mêmes

poids, les approches SK-means et RSKC pourraient ne pas nécessairement fonctionner mieux que les approches K-means et TK-means. Les approches SK-means RSKC sont spécifiquement conçues pour gérer des données avec des variables ayant des poids différents. Elles attribuent des poids à ces variables et utilisent une approche parcimonieuse pour sélectionner les plus importantes. Si les variables ont la même importance et ne nécessitent pas de poids différents, alors l'avantage principal du SK-means et du RSKC, qui est la gestion des caractéristiques pondérées, pourrait ne pas se manifester. La figure 5.4.4 illustre cette situation ; on peut voir à peine deux poids importants et le reste très proches de zéro.

Les résultats obtenus dans cette application nous amènent à conclure qu'il serait prudent avant de se pencher sur une méthode sans évaluer les autres surtout quand on ne connaît pas parfaitement la structure de nos données. Le choix d'une méthode dépend donc du contexte et de la nature des données.

Dans le cadre de cette étude, bien que les résultats encourageants aient été obtenus, il est important de reconnaître les limites à la méthodologie et aux choix effectués. Ces limitations offrent des opportunités pour des recherches futures plus approfondies et pourraient influencer l'interprétation des résultats obtenus. Les principales limites identifiées sont les suivantes :

La limite liée à l'exploration d'autres bases de données textuelles : L'utilisation exclusive de la base de données "20 Newsgroups" a fourni des résultats significatifs en ce qui concerne le regroupement des documents en classes, mais pas pour l'objectif visé dans cette étude qui était de promouvoir les méthodes parcimonieuses. Cependant, il convient de noter que cette base de données est spécifique et ne couvre qu'une gamme limitée de sujets. Pour évaluer la généralisation de nos conclusions, il serait judicieux d'étendre cette analyse à d'autres bases de données textuelles avec des contextes et des domaines de sujets différents. L'exploration de diverses sources de données textuelles pourrait révéler des tendances et des structures différentes, ce qui pourrait renforcer les conclusions obtenues ici.

La limite liée à l'exploration d'autres mesures de similarité : Dans cette étude, nous avons principalement utilisé la distance euclidienne comme mesure de similarité pour les algorithmes de regroupement. Cependant, il existe une variété de mesures de similarités alternatives, telles que la similarité cosinus qui est très utilisée en NLP, la distance de Jaccard ou la divergence de Kullback-Leibler. L'expérimentation avec différentes mesures de similarité pourrait fournir des perspectives supplémentaires sur les relations entre les documents et les différentes classes, et potentiellement conduire à des résultats différents.

Limite quant à l'exploration d'autres algorithmes : Bien que cette étude ait employé une sélection d'algorithmes de regroupement populaires, tels que le K-means, le Trimmed K-means, le SK-means et le RSK-means, il existe d'autres méthodes de regroupement disponibles utilisant d'autres types de pénalisations tels que "Elastic Net". L'exploration d'algorithmes alternatifs tels que le DBSCAN, le weighted K-medoid ou le entropy-weighted K-means pourraient révéler des structures ou des relations différentes entre les documents.

Ces limites invitent à la prudence lors de l'interprétation des conclusions et soulignent l'importance de prendre en compte ces aspects pour une compréhension approfondie des résultats obtenus.

Les méthodes parcimonieuses se présentent comme des outils prometteurs pour la découverte de structures cachées dans des ensembles de données complexes telles que les données textuelles. Dans des recherches futures, il serait intéressant de poursuivre cette exploration en y intégrant les méthodes de vectorisation dense utilisant le réseau de neurones introduit dans l'article [32], connu sous le nom de Word2Vec ou Doc2Vec qui pourraient ainsi contribuer à améliorer les résultats obtenus.

Bibliographie

- [1] T. D. S. S. Hrouda-Rasmussen, “An intuitive demonstration of the curse of dimensionality, and what we can do to combat it.” <https://towardsdatascience.com/the-curse-of-dimensionality-5673118fe6d2>. Accessed : 2023-06-2.
- [2] L. Parsons, E. Haque, and H. Liu, “Subspace clustering for high dimensional data : a review,” *Acm sigkdd explorations newsletter*, vol. 6, no. 1, pp. 90–105, 2004.
- [3] S. Naeem and A. Wumaier, “Study and implementing k-mean clustering algorithm on english text and techniques to find the optimal value of k,” vol. 182, no. 31, pp. 7–14.
- [4] R. Bellman, “Dynamic programming, princeton univ,” *Press Princeton, New Jersey*, 1957.
- [5] J. Hua, Z. Xiong, J. Lowey, E. Suh, and E. R. Dougherty, “Optimal number of features as a function of sample size for various classification rules,” *Bioinformatics*, vol. 21, pp. 1509–1515, 11 2004.
- [6] S. Karanam, “Curse of dimensionality, a “curse” to machine learning.” <https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb>. Accessed : 2023-06-2.
- [7] G. Hughes, “On the mean accuracy of statistical pattern recognizers,” *IEEE transactions on information theory*, vol. 14, no. 1, pp. 55–63, 1968.
- [8] J. Lee, “Information visualisation.” <https://moodle.uclouvain.be/course/view.php?id=3502>, 2020. Accessed : 2023-05-15.
- [9] N. Venkat, “The curse of dimensionality : Inside out,” *Pilani (IN) : Birla Institute of Technology and Science, Pilani, Department of Computer Science and Information Systems*, vol. 10, 2018.
- [10] J. Segers, “Analyse des données.” <https://moodle.uclouvain.be/course/view.php?id=1225>, 2020. Accessed : 2023-05-15.
- [11] A. L. Levada, “Pca-kl : a parametric dimensionality reduction approach for unsupervised metric learning,” *Advances in Data Analysis and Classification*, vol. 15, no. 4, pp. 829–868, 2021.
- [12] D. M. Witten and R. Tibshirani, “A framework for feature selection in clustering,” *Journal of the American Statistical Association*, vol. 105, no. 490, pp. 713–726, 2010.

- [13] W.-C. Chang, "On using principal components before separating a mixture of two multivariate normal distributions," *Journal of the Royal Statistical Society : Series C (Applied Statistics)*, vol. 32, no. 3, pp. 267–275, 1983.
- [14] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.
- [15] W. Pan and X. Shen, "Penalized model-based clustering with application to variable selection," *Journal of machine learning research*, vol. 8, no. 5, 2007.
- [16] A. E. Raftery and N. Dean, "Variable selection for model-based clustering," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 168–178, 2006.
- [17] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.
- [18] M. Bhatia and D. Khurana, "Analysis of initial centers for k-means clustering algorithm," *International journal of computer applications*, vol. 71, no. 5, 2013.
- [19] T. M. Kodinariya, P. R. Makwana, *et al.*, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.
- [20] K. S. Pollard and M. J. Van Der Laan, "A method to identify significant clusters in gene expression data," 2002.
- [21] L. Kaufman and P. J. Rousseeuw, *Finding groups in data : an introduction to cluster analysis*. John Wiley & Sons, 2009.
- [22] Halkidi, B. Maria, V. Yannisi, and Michailis, "On clustering validation techniques," 2001.
- [23] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.
- [24] R. V. Singh and M. S. Bhatia, "Data clustering with modified k-means algorithm," in *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pp. 717–721, IEEE, 2011.
- [25] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data : A survey on subspace clustering, pattern-based clustering, and correlation clustering," *Acm transactions on knowledge discovery from data (tkdd)*, vol. 3, no. 1, pp. 1–58, 2009.
- [26] J. A. Cuesta-Albertos, A. Gordaliza, and C. Matrán, "Trimmed k -means : an attempt to robustify quantizers," *The Annals of Statistics*, vol. 25, no. 2, pp. 553–576, 1997.
- [27] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.
- [28] J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, "Bayesian clustering with variable and transformation selections," in *Bayesian Statistics 7 : Proceedings of the Seventh Valencia International Meeting*, vol. 249, Oxford University Press, USA, 2003.

-
- [29] Y. Kondo, M. Salibian-Barrera, and R. Zamar, “Rskc : An r package for a robust and sparse k-means clustering algorithm,” *Journal of Statistical Software*, vol. 72, pp. 1–26, 2016.
 - [30] J. H. M. Daniel Jurafsky, “An introduction to natural language processing, computational linguistics, and speech recognition,” pp. 10–35, 2018.
 - [31] J. H. M. Daniel Jurafsky, “An introduction to natural language processing, computational linguistics, and speech recognition,” pp. 37–130, 2018.
 - [32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv :1301.3781*, 2013.

Annexe A

Code R du chapitre 4

```
1 # packages used
2 library(MASS)
3 library(ggplot2)
4 require(plotly)
5 require(RSKC)
6 #library(FactoMineR)
7 require(patchwork)
8 require(subspace)
9 library(sparcl)
10 require(kableExtra)
11 theme_set(theme_bw())
12 #####
13 #Function for simulating our data.
14 #####
15 # Set parameters
16 sim<-function(n,p,m,K,sd=1){
17 #n number of observations
18 #p total number of features
19 #K number of clusters
20 #m number of clustering features
21 sd standard deviation for multivariate normal distribution
22 # Generate cluster mean vectors
23 #e=rep(1, m)
24 #f=p-m
25 #d=rep(0,f)
26 b <- c(rep(1, m),rep(0, p-m)) # vector b as described in the prompt
27 mu <- c(-2, 0, 2) # vector of mean values for each cluster
28 means <- t(outer(mu, b)) # matrix of mean vectors, with each row representing a cluster
29 cluster_size=n/K
30 # Generate data
31 set.seed(10) # set seed for reproducibility
32 dat <- matrix(NA, nrow = n, ncol = p)
33 for (i in 1:K){
34   start<-(i-1)*cluster_size + 1
35   end <- i*cluster_size
36   dat[start:end,]<-mvrnorm(cluster_size , mu = means[i,], Sigma =diag(p)*sd^2)
```

```

37 # print(diag(p)*sd^2)
38 }
39 dat=data.frame(dat)
40
41 # Add cluster labels
42 dat$cluster_labels<-rep(1:K, each = n/K)
43 #dat <- cbind(cluster_labels, dat)
44 dat$cluster_labels<-factor(dat$cluster_labels)
45 return (dat)
46 }
47
48 # Function to define outliers
49 abera<-function(dat){
50   variables <- c("X1", "X2")
51
52   dt<-dat
53
54   valeurs_aberrantes <- list()
55   for (var in variables) {
56     #mean_var <- mean(dt[[var]])-3
57     #sd_var <- sd(dt[[var]])
58     valeurs_aberrantes[[var]] <-runif(10,min=-19,max=-10)
59     #rnorm(10,mean=mean_var ,sd=sd_var)
60   }
61 }
62
63 # add outliers to dataset
64 for (var in variables) {
65   index_aberrant <-500:519
66   dt[index_aberrant, var] <- valeurs_aberrantes[[var]]
67 }
68 return(dt)
69 }
70 #####
71 # Data simulation without outliers
72 #####
73 dat<-sim(1200,600,2,3)
74 data0<-dat |> subset(select=-cluster_labels)
75 #Scatter plot graph by combining a few variables.
76 pairs(
77   dat[,c(1,2,3,60)],col=dat$cluster_labels,
78   lower.panel=NULL
79 )
80
81 # Plot of true labels in 2 dimension
82 p0<-ggplot(dat, aes(x = X1, y = X2, color =cluster_labels, shape = cluster_labels))
83   geom_point(size = 2) +
84   labs(title="true cluster labels",x = "Feature 1", y = "Feature 2",
85     color = "Cluster", shape = "Cluster")
86 # Applying and plot K-mrans,Trimmed K-means,SK-means and RSKC
87 res01 <- kmeans(data0, centers = 3)

```

```

88 data01<-data0
89 data01$cluster<-(res01$cluster)
90 data01$cluster<-factor(data01$cluster)
91 p01<-ggplot(data01, aes(x = X1, y = X2, color =cluster, shape = cluster)) +
92   geom_point(size = 2) +
93   labs(x = "Feature 1", y = "Feature 2", color = "Cluster", shape = "Cluster",
94        title="k-mean")
95 # SK=means
96 res02<-KMeansSparseCluster(data0,3,wbounds=6)
97
98 data02<-data0
99 data02$cluster<-res02[[1]]$Cs
100 data02$cluster<-factor(data02$cluster)
101 p02<-ggplot(data02, aes(x = X1, y = X2, color =cluster, shape = cluster)) +
102   geom_point(size = 2) +
103   labs(title="sk-mean",x = "Feature 1", y = "Feature 2", color = "Cluster",
104        shape = "Cluster")
105
106 res03<-RSKC(data0, L1 =6, nstart =30, ncl = 3, alpha = 0.1)
107 # RSKC
108 data03<-data0
109 data03$cluster<-sort(res03$labels)
110 data03$cluster<-factor(data03$cluster)
111 p03<-ggplot(data03, aes(x = X1, y = X2, color =cluster, shape = cluster)) +
112   geom_point(size = 2) +
113   labs(title="RSKC",x = "Feature 1", y = "Feature 2", color = "Cluster",
114        shape = "Cluster")
115 # Trimmed K-means
116 res04<-RSKC(data0, L1 =NULL, nstart =30, ncl = 3, alpha = 0.1)
117 data04<-data0
118 data04$cluster<-sort(res04$labels)
119 data04$cluster<-factor(data04$cluster)
120 p04<-ggplot(data04, aes(x = X1, y = X2, color =cluster, shape = cluster)) +
121   geom_point(size = 2) +
122   labs(title="Trimmed K-means",x = "Feature 1", y = "Feature 2",
123        color = "Cluster", shape = "Cluster")
124 # Plot all methods
125 (p0 +p01+p02+p04+p03)+
126   plot_layout(nrow = 3)
127 #####
128 # With outliers: Use the same code above (change data0 to data1
129 #####
130 dt<-abera(dat)
131 data1<-dt|> subset(select=-cluster_labels)
132
133 #####
134 Case A n>>p: without outliers
135 #####
136 sim1<-sim(1200,600,50,3)
137 sim11<-sim1 |> subset(select=-cluster_labels)
138 # choose tuning parameter

```

```

139 km.perm <- KMeansSparseCluster.permute(sim11,K=3,wbounds=seq(3,9,len=15),
140 nperms=4)
141 print(km.perm)
142 plot(km.perm)
143 # run sparse k-means
144 km.out <- KMeansSparseCluster(sim11,K=3,wbounds=km.perm$bestw)
145 print(km.out)
146 plot(km.out)
147 cat("best L1",km.perm$bestw)
148
149 #RSKC
150 alphas <- c(0.05, 0.1,0.2)
151 l.a <- length(alphas)
152 minL1 <- 2
153 maxL1 <- 16
154 #L1NULL <- maxL1 + 1.5
155 Lis <- seq(minL1, maxL1,2)
156 l.L <- length(Lis)
157 nonZero <- clustSize <- reCER <- matrix(NA, l.a, l.L)
158 temp <- rep(list(NULL), l.L)
159 names(temp) <- paste("L1", Lis, sep = "")
160 re.rskc <- rep(list(temp), l.a)
161 names(re.rskc) <- paste("alpha", alphas, sep = "")
162 set.seed(1)
163 for (ialpha in 1:l.a) {
164   for (iL in 1:l.L) {
165     cat("\nAlpha", alphas[ialpha], "L1", Lis[iL])
166     #if (Lis[iL] == L1NULL)
167       #L1 <- NULL else
168     L1 <- Lis[iL]
169     re.rskc[[ialpha]][[iL]] <- rskc <- RSKC(sim11, L1 = L1, ncl = 3, alpha = alphas[ialpha],
170       nstart = 10, silent = TRUE)
171     print(rskc)
172     reCER[ialpha, iL] <- CER(rskc$label,sim1$cluster_labels)
173     clustSize[ialpha, iL] <- min(sum(rskc$label == 1), sum(rskc$label == 2))
174     nonZero[ialpha, iL] <- sum(rskc$weight > 0)
175   }
176 }
177 # Comparison
178 # K-means
179 cer_k<-c()
180 for(i in 1:100){
181   sampl<-sample(sim11,replace=TRUE)
182   k.out<-kmeans(sampl, centers = 3)
183   boot_k<-CER(k.out$cluster,true.ind=sim1$cluster_labels)
184   cer_k<-c(cer_k,boot_k)
185 }
186 # SK-means
187 cer_sk<-c()
188 for(i in 1:100){
189   sampl<-sample(sim11,replace=TRUE)

```

```

190 # sk.out<-KMeansSparseCluster(sampl,K=3,wbounds=3)
191 boot_sk<-CER(km.out[[1]]$Cs,true.ind=sim1$cluster_labels)
192 cer_sk<-c(cer_sk,boot_sk)
193
194 }
195 # RSKC
196 cer_rsk<-c()
197 for(i in 1:100){
198   sampl<-sample(sim11,replace=TRUE)
199   rsk.out<-RSKC(sampl, ncl = 3, alpha = 0.05, L1 =6, nstart = 10)
200   boot_rsk<-CER(rsk.out$labels,true.ind=sim1$cluster_labels)
201   cer_rsk<-c(cer_rsk,boot_rsk)
202 }
203
204 # Trimmed K-means
205 cer_Tk<-c()
206 for(i in 1:100){
207   sampl<-sample(sim11,replace=TRUE)
208   rsk.out<-RSKC(sampl, ncl =NULL, alpha = 0.1, L1 =6, nstart = 10)
209   boot_Tk<-CER(Tk.out$labels,true.ind=sim1$cluster_labels)
210   cer_Tk<-c(cer_Tk,boot_Tk)
211 }
212
213 #####
214 Case A n>>p: withoutliers
215 #####
216 sim2<-abera(sim1)
217 sim22<-sim2|> subset(select=-cluster_labels)
218 # use the same code above.
219 #####
220 #For the case n<<p, the same code is used by varying the #parameters.

```

Annexe B

Code python du chapitre 5

```
1 # Installing package
2 from nltk.stem import PorterStemmer
3 from nltk.stem import WordNetLemmatizer
4 import seaborn as sns
5
6 import re
7 import pickle
8 import joblib
9 #from sklearn.externals import joblib
10
11 import os
12 import math
13 # Data Structures
14 import numpy as np
15 import pandas as pd
16 import geopandas as gpd
17 import json
18
19 # Corpus Processing
20 import nltk.corpus
21 from unicode import unicode
22 from nltk.tokenize import word_tokenize
23 from nltk import SnowballStemmer
24 from sklearn.feature_extraction.text import TfidfVectorizer
25 from sklearn.preprocessing import normalize
26
27 # K-Means
28 from sklearn import cluster
29
30 # Visualization and Analysis
31 import matplotlib.pyplot as plt
32 import matplotlib.cm as cm
33 import seaborn as sns
34 from sklearn.metrics import silhouette_samples, silhouette_score
35 from wordcloud import WordCloud
36
```

```
37 # Map Viz
38 import folium
39 #import branca.colormap as cm
40 from branca.element import Figure
41 from nltk.corpus import stopwords
42 nltk.download('stopwords')
43 from sklearn.cluster import KMeans
44 # dataset
45 from sklearn.datasets import fetch_20newsgroups
46
47 categories = ['soc.religion.christian',
48               'comp.sys.ibm.pc.hardware',
49               'talk.politics.guns'
50               ]
51 # Load Data
52 twenty_train = fetch_20newsgroups(subset='train', categories=categories,
53 shuffle=True, random_state=42)
54 twenty_test = fetch_20newsgroups(subset='test', categories=categories,
55 shuffle=True, random_state=42)
56
57 # Check number of records in training and testing data
58 len(twenty_train.data), len(twenty_test.data)
59 df = pd.DataFrame()
60 df['text'] = twenty_train.data
61 df['target'] = twenty_train.target
62 df['target'] = df['target'] + 1
63 df.head()
64
65 #Visulaize numbers of Category of data
66 df.groupby('target').target.value_counts().plot(kind = "bar",
67 color = ["red", "pink", "blue"])
68 plt.xlabel("Categories des donnees")
69 plt.title("Visulaize numbers of Category of data")
70 plt.show()
71 #Histogram of the articles number of characters
72 plt.figure(figsize=(10,8))
73 ax = sns.distplot(df.text_count.apply(math.log))
74 ax.set(title='Histogram of the articles number of characters log transformed.',
75 xlabel='Log of Article length', ylabel='Proportions')
76 plt.show()
77
78 print('There are in total {} articles containing a text body with their lengths norm
79 distributed around the average length of: {} words.'.format(len(df.text_count), round
80 df['text_count'] = df['text'].apply(lambda x: len(x.strip().split()))
81
82 df.head()
83 #####
84 #Preprocessing
85 #####
86 def remove_tags(text):
87     remove = re.compile(r'')
```

```
88     return re.sub(remove, '', text)
89 df['Text'] = df['text'].apply(remove_tags)
90 df['Text'][0]
91
92 def remove_wordnet(text):
93     return re.sub("\S*\S*\s?", " ", text)
94 df['Text'] = df['Text'].apply(remove_wordnet)
95 df['Text'][0]
96
97 def remove_number(text):
98     return re.sub("\S*\d\S*", " ", text)
99 df['Text'] = df['Text'].apply(remove_number)
100 df['Text'][0]
101 def special_char(text):
102     reviews = ''
103     for x in text:
104         if x.isalnum():
105             reviews = reviews + x
106         else:
107             reviews = reviews + ' '
108     return reviews
109 df['Text'] = df['Text'].apply(special_char)
110 df['Text'][0]
111
112 def convert_lower(text):
113     return text.lower()
114 df['Text'] = df['Text'].apply(convert_lower)
115 df['Text'][0]
116
117 import nltk
118 nltk.download('punkt')
119 def remove_stopwords(text):
120     stop_words = set(stopwords.words('english'))
121     words = word_tokenize(text)
122     return [x for x in words if x not in stop_words]
123 df['Text'] = df['Text'].apply(remove_stopwords)
124
125 nltk.download('wordnet')
126 def lemmatize_word(text):
127     wordnet = WordNetLemmatizer()
128     return " ".join([wordnet.lemmatize(word) for word in text])
129 df['Text'] = df['Text'].apply(lemmatize_word)
130 df['Text'][0]
131
132 # TF-IDF Feature Generation
133 from sklearn.feature_extraction.text import TfidfVectorizer
134 from nltk.tokenize import RegexpTokenizer
135
136 # Initialize regex tokenizer
137 tokenizer = RegexpTokenizer(r'\w+')
138
```

```
139
140 # # Vectorize document using TF-IDF
141 tf_idf_vect = TfidfVectorizer(lowercase=True,
142                               #stop_words='english',
143                               ngram_range = (1,1),
144                               tokenizer = tokenizer.tokenize,
145                               max_df=0.5,
146                               min_df=5)
147
148 # Fit and Transform Text Data
149 X_train_counts = tf_idf_vect.fit_transform(df['Text'])
150
151 # Check Shape of Count Vector
152 X_train_counts.shape
153
154
155 tf_idf = pd.DataFrame(data = X_train_counts .toarray(),
156                       columns=tf_idf_vect.get_feature_names_out())
157
158 final_df0 = tf_idf
159
160 print("{} rows".format(final_df0.shape[0]))
161 final_df0.T.nlargest(5, 0)
162
163 # Create Kmeans object and fit it to the training data
164 kmeans = KMeans(n_clusters=3,random_state=42).fit(X_train_counts.toarray())
165
166 # Get the labels using KMeans
167 pred_labels = kmeans.labels_
168 pred_labels1 = np.add(pred_labels, 1)
169 pred_labels1
170
171 # Applying t-SNE
172
173 from sklearn.manifold import TSNE
174
175 tsne = TSNE(verbose=1, perplexity=50) # Changed perplexity from 100 to 50 per FAQ
176 X_embedded = tsne.fit_transform(X_train_counts.toarray())
177 sns.set(rc={'figure.figsize':(15,15)})
178
179 # colors
180 palette = sns.color_palette("bright", 1)
181
182 # plot
183 sns.scatterplot(x=X_embedded[:,0], y=X_embedded[:,1], palette=palette)
184 plt.title('t-SNE with no Labels')
185 plt.savefig("t-sne_covid19.png")
186 plt.show()
187
188 # sns settings
189 sns.set(rc={'figure.figsize':(13,9)})
```

```
190
191 # colors
192 palette = sns.hls_palette(3, l=.4, s=.9)
193
194 # plot
195 sns.scatterplot(x=X_embedded[:,0], y=X_embedded[:,1], hue=pred_labels1,
196 legend='full', palette=palette)
197 plt.title('t-SNE with Kmeans Labels')
198 plt.savefig("improved_cluster_tsne.png")
199 plt.show()
200
201 # Import WordCloud and STOPWORDS
202 from wordcloud import WordCloud
203 from wordcloud import STOPWORDS
204 # Import matplotlib
205 import matplotlib.pyplot as plt
206 import numpy as np # linear algebra
207
208
209 def word_cloud(text, wc_title, wc_file_name='wordcloud.jpeg'):
210     # Create stopword list
211     stopword_list = set(STOPWORDS)
212
213     # Create WordCloud
214     word_cloud = WordCloud(width = 800, height = 500,
215                             background_color = 'white',
216                             stopwords = stopword_list,
217                             min_font_size = 14).generate(text)
218
219     # Set wordcloud figure size
220     plt.figure(figsize = (8, 6))
221
222     # Set title for word cloud
223     # plt.title(wc_title)
224
225     # Show image
226     plt.imshow(word_cloud)
227
228     # Remove Axis
229     plt.axis("off")
230
231     # save word cloud
232     plt.savefig(wc_file_name, bbox_inches='tight')
233
234     # show plot
235     plt.show()
236
237 def get_top_features_cluster(tf_idf_array, prediction, n_feats):
238     labels = np.unique(prediction)
239     dfs = []
240     for label in labels:
```

```

241         id_temp = np.where(prediction==label) # indices for each cluster
242         x_means = np.mean(tf_idf_array[id_temp], axis = 0) # returns average score a
243         sorted_means = np.argsort(x_means)[::-1][:n_feats] # indices with top 20 sc
244         features = tf_idf_vect.get_feature_names_out()
245         best_features = [(features[i], x_means[i]) for i in sorted_means]
246         df = pd.DataFrame(best_features, columns = ['features', 'score'])
247         dfs.append(df)
248     return dfs
249 def plotWords(dfs, n_feats):
250     plt.figure(figsize=(8, 4))
251     for i in range(0, len(dfs)):
252         plt.title(("Most Common Words in Cluster {}".format(i)), fontsize=10,
253                 fontweight='bold')
254         sns.barplot(x = 'score' , y = 'features', orient = 'h' , data = dfs[i][:n_feats])
255     plt.show()
256     #####
257     # The RSKC and SPARCL packages not being available in Python, we loaded
258     # the package that enables the use of the R language in Google Colab alongside Python
259     # and utilized the same functions as in Chapter 4.
260     #####

```

