Science des données

IFT799 – TP1

Amazon Book Reviews

UNIVERSITÉ DE SHERBROOKE

Elèves : Tahar AMAIRI et Corentin POMMELEC

Session : automne 2022

Introduction :

Le code du TP est contenu dans un Jupyter Notebook. Pour son exécution, il nécessite plusieurs modules déjà très connus dans le domaine de la science des données : pandas, numpy, matplotlib etc... Cependant certains ne le sont pas comme IPython et imageio : ces librairies peuvent être facilement installées à l'aide de "pip".

Le jeu de données que nous utilisons est celui de l'Amazon Book Reviews (ABR) contenant les revues données par les clients sur les livres qu'ils ont achetés sur Amazon. Celui-ci se trouve dans un fichier JSON contenant différentes informations sur les avis des clients qui vont nous être indispensables pour réaliser notre analyse descriptive. On y trouve par exemple l'ASIN (Amazon Standard Identification Number) du livre, sa note ou bien des indications temporelles sur l'avis (unix time, raw time).

Pour charger le fichier JSON contenant toutes les reviews, nous allons le découper en "chunk" afin de pouvoir le lire car il est extrêmement lourd (~10 GB). Pour cela, il est possible en fonction de la puissance de votre ordinateur d'augmenter la taille des chunks et/ou leurs nombres. Dans le cadre de la rédaction de ce rapport, nous avons chargé 1 million d'avis et conseillons à notre lecteur ce nombre s'il possède un ordinateur portable (~1min 30s de temps de chargement).

a.1) Quelle est la moyenne de score de chaque livre ?

Afin d'obtenir l'appréciation d'un livre, il est nécessaire de calculer sa moyenne pondérée. Pour cela il suffit de multiplier chaque score par son nombre de review et de diviser par le nombre d'avis total. Cette question est implémentée par la fonction `getAverageScore` : celle-ci enregistre dans un dictionnaire pour chaque livre la moyenne de score et le nombre d'avis. Concernant les calculs, on utilise le produit scalaire de numpy entre le vecteur [1,2,3,4,5] et chaque colonne de la matrice des scores puis le résultat est divisé par la somme de cette dite colonne. Par la suite, ce dictionnaire est transformé en un "data frame" :

	000100039X	0001055178	0001473123	0001473727	0001473905
Mean score	4.674757	3.555556	4.625	5.0	4.666667
Number of ratings	206.000000	18.000000	16.000	7.0	6.000000

On remarque tout de suite le problème posé par l'utilisation de la moyenne : est-ce que le livre "0001473727" est-il vraiment meilleur que "000100039X" avec ses 7 avis alors que le second possède 206 avis ? On discutera d'une alternative ultérieurement pour comparer deux livres en prenant en compte le nombre de reviews.

a.2) Quels sont le(s) livre(s) le(s) mieux apprécié(s) et le(s) moins apprécié(s) ?

Maintenant que nous avons la moyenne de score de chaque livre on peut facilement les trier dans l'ordre que l'on souhaite à l'aide de la fonction pandas "sort_values" : dans notre cas, on les a classés dans l'ordre croissant. Pour sélectionner les "n" meilleurs/mauvais livres, nous avons implémenté deux fonctions : `getWorstBooks` et `getBestBooks`.

Chacune d'elle utilise la fonction pandas "iloc" pour pouvoir sélectionner les n premières (mauvais livres) ou dernières (meilleurs livres) colonnes. Pour le dernier cas, on utilise une indexation négative car on a trié de manière croissante.

	0140189882	0140266313	015200470X	0062043935
Score	5.0	5.0	5.0	5.0
Number of ratings	5.0	6.0	6.0	5.0

4 premiers meilleurs livres

	0071481478	006000455X	0071393080	006052846X
Mean score	1.0	1.214286	1.333333	1.354839
Number of ratings	6.0	14.000000	6.000000	31.000000

4 premiers mauvais livres

a.3) Quels est le 1er quart des livres les plus appréciés ?

L'implémentation de cette fonction se trouve dans la fonction getFirstQuarterBestBooks :

Celle-ci utilise la même idée que pour la question précédente. Cependant, ici l'indexation négative se base sur le nombre de livres multiplié par 0.25 afin d'obtenir un quart de livres.

4) Entre deux livres, lequel est mieux apprécié ?

Pour savoir entre deux livres lequel est le plus apprécié uniquement à partir des opérations précédentes, que nous avons effectuées sur les scores obtenus par les livres, le plus évident est de comparer directement la moyenne pondérée obtenue pour chacun de ces livres. En récupérant deux livres aléatoirement dans le data frame contenant la moyenne pondérée (Mean score) de chaque livre, nous pouvons comparer la valeur de leur moyenne. Notre fonction getBestOfTwoBooks affichera le livre avec la moyenne la plus élevée ou avertira l'utilisateur si les moyennes sont égales. Le livre est ici considéré comme le plus apprécié par rapport à la seule opération que l'on a effectué jusqu'alors : la moyenne pondérée.

```
0062075551 has a better score than 0062286706
- 0062075551 : score = 4.833333333333333, number of reviews = 6.0
- 0062286706 : score = 4.553191489361702, number of reviews = 94.0
```

Exemple d'affichage de la comparaison entre 2 livres

5) Est-ce que l'utilisation des comparaisons de scores moyennes est toujours une bonne façon de faire pour répondre à ces questions ? Sinon, quelles sont les alternatives ?

Cependant, l'utilisations des comparaisons de scores moyennes n'est pas toujours la bonne façon de faire pour répondre à ces questions notamment lors de la comparaison de l'appréciation. En effet, par exemple, un livre avec peu d'avis et la note maximale ne devrait pas forcément être considéré comme

plus apprécié qu'un livre avec énormément d'avis et une note proche du maximum. En outre, ici les scores sont fixés entre 1 et 5 mais dans le cas de valeurs qui peuvent être extrêmes, comme vu en cours avec l'exemple des salaires, la moyenne n'est pas assez robuste et les valeurs extrêmes crées un biais positif ou négatif dans le résultat. Dans le cas cité précédemment, on peut par exemple penser à l'utilisation de la médiane, c'est-à-dire le point milieu d'un jeu de données, qui est plus robuste aux valeurs extrêmes. Pour en revenir à notre cas particulier de la comparaison d'appréciation, la moyenne bayésienne, ou bayesian average en anglais, correspond à nos attentes. Cette dernière est une méthode d'estimation de la moyenne qui utilise des informations extérieures, principalement basées sur une croyance existante, qui sont prises en compte dans le calcul. La bayesian average nous permettra via l'utilisation de ces informations extérieures d'ajuster les moyennes des scores des livres en fixant un seuil de fiabilité sur le nombre d'avis des livres. Ainsi, le nombre d'avis deviendra un des facteurs d'importance au même titre que la moyenne des scores du livre.

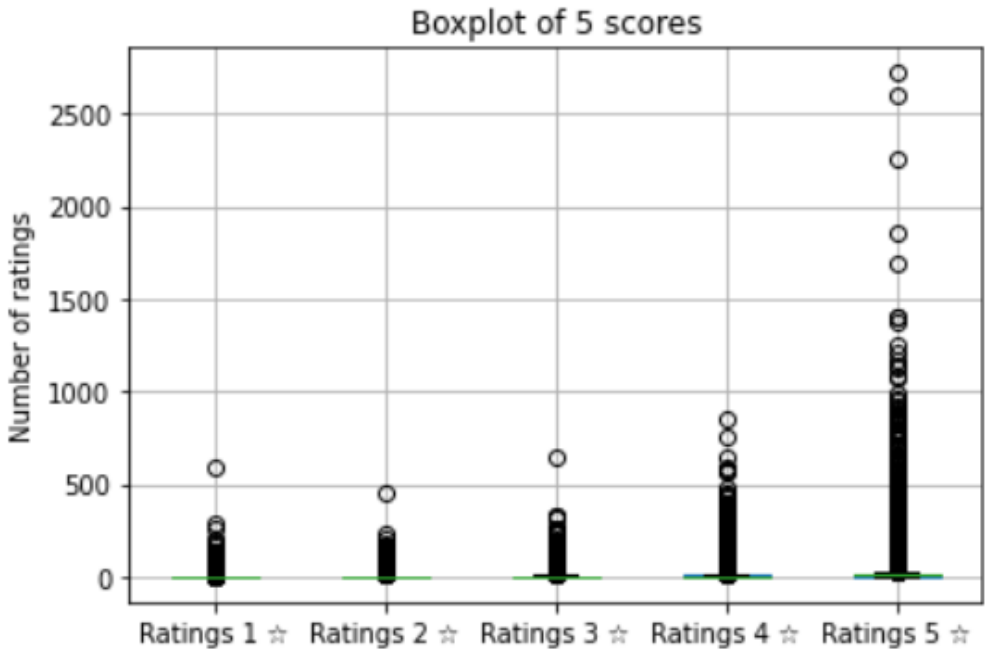
Nous calculerons la moyenne bayésienne sur chaque livre de la matrice lors de la projection des livres dans le plan de l'ACP via la fonction getBayesianScore. Le calcul utilisé est le suivant :

$$\bar{x} = \frac{C \times m + \ Score \ \times nombre\ de\ note}{C \times nombre\ de\ note}$$

C : le nombre d'avis fixé comme seuil de confiance

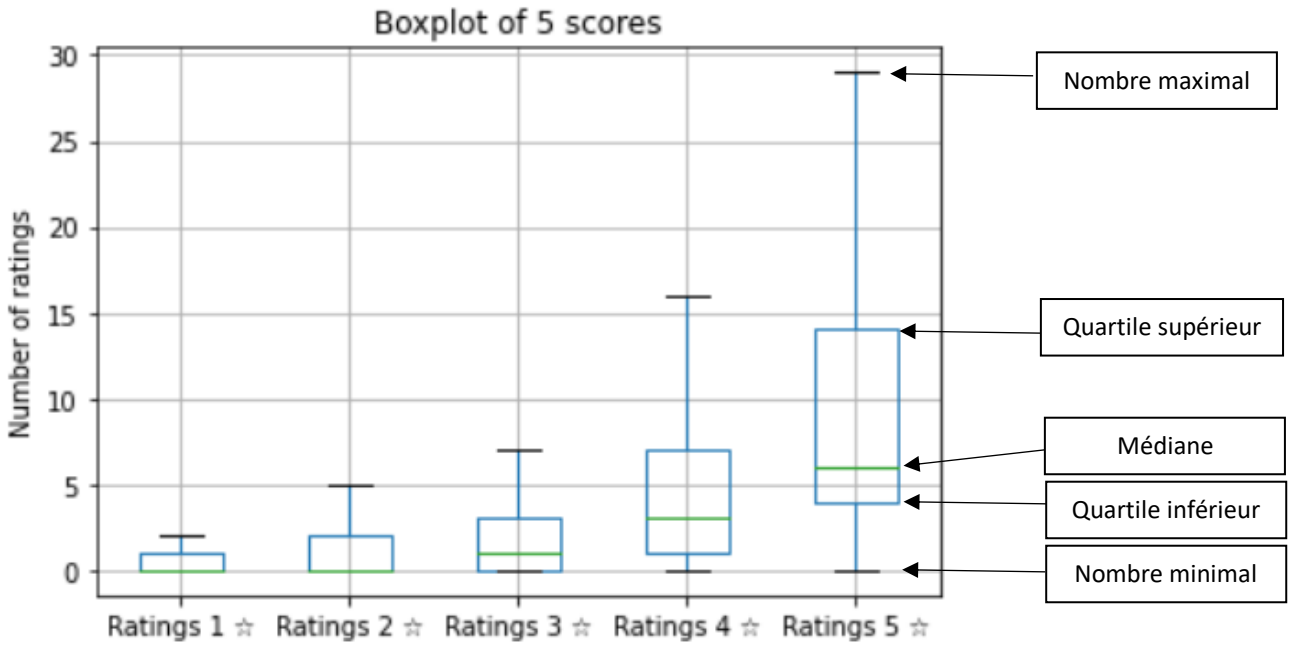
m : A priori sur la moyenne des notes

6) A l'aide de la bibliothèque python Matplotlib permettant de tracer et de visualiser des données sous forme graphique nous traçons un diagramme en moustache, traduit box plot en anglais. Nous utiliserons ce dernier afin d'afficher les tendances statistiques pour chacun des scores 1, 2, 3, 4, 5 à partir du data frame initial indiquant le nombre d'avis par score pour chaque livrés étudiés. Le premier affichage du diagramme sans modification est trop étendu à cause de nombreuses valeurs aberrantes allant jusqu'à près de 3000 avis. Cela a pour conséquence une très mauvaise visibilité des box plots, ce qui rend leur analyse impossible.



Box plot avec outliers

La solution retenue est de retirer l'affichage des valeurs aberrantes pour que le diagramme soit centré sur les box plots à analyser :



Box plot avec outliers

A partir de ce diagramme il est désormais possible d'analyser les différents box plots.

	Score = 1	Score = 2	Score = 3	Score = 4	Score = 5
Nombre minimal	0	0	0	0	0
Nombre maximale	2	5	7	16	29
Médiane	0	0	1	3	6
Quartile inférieur	0	0	0	1	3
Quartile supérieur	1	2	3	7	14

Nous pouvons voir que les boites à moustaches permettent d'obtenir un résumé visuel des données afin d'identifier rapidement les valeurs médianes, la dispersion de l'ensemble de données et montre les signes d'asymétrie. Pour aider l'analyse, les boites à moustache divisent les données en sections contenant chacune environ 25 % des données de l'ensemble. Les sections sont entre la valeur minimale et le quartile inférieur, le quartile inférieur et la médiane, la médiane et le quartile supérieur et entre le quartile supérieur et la valeur maximale.

Dans notre cas nous pouvons nous rendre compte qu'il y a une asymétrie en haut sur tous les box plots. On en déduit que le nombre d'avis par livre est faible, soit les lecteurs ne prennent pas le temps de noter le livre soit de nombreux livres ne sont pas lu par beaucoup de monde. Cela se vérifie via toutes les autres métriques avec des médianes variant de 0 à 6 notes par exemple.

On remarque également que plus le score est élevé plus le nombre de note est élevé. On en déduit que les livres ont tendances à être bien notés.

Enfin, la dispersion permet de confirmer ces analyses, par exemple on a 25% du nombre de notes supérieur entre 14 et 29, pour le score 5 contre 25% du nombre de notes supérieur entre 1 et 2 pour le score 1.

b.1) Faire une analyse en composantes principales pour représenter chaque livre par la projection sur les deux premières composantes principales et afficher le nuage de points représentant les livres sur le plan.

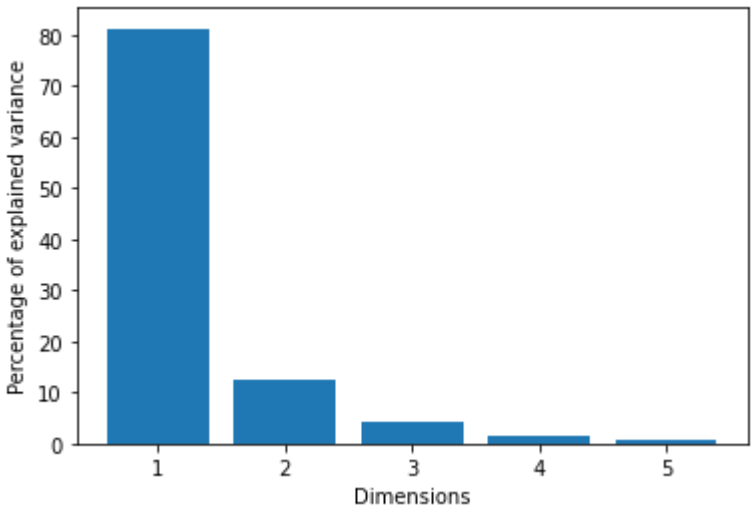
Afin de calculer une ACP, il est nécessaire de suivre différentes étapes :

1. Normaliser les données
2. Calculer la matrice de covariance de ces données
3. Calculer les valeurs propres et les vecteurs propres associés de cette matrice de covariance

Etant donné que nous utilisons des DataFrame de pandas et le module numpy, il est très simple d'effectuer chacune de ces étapes :

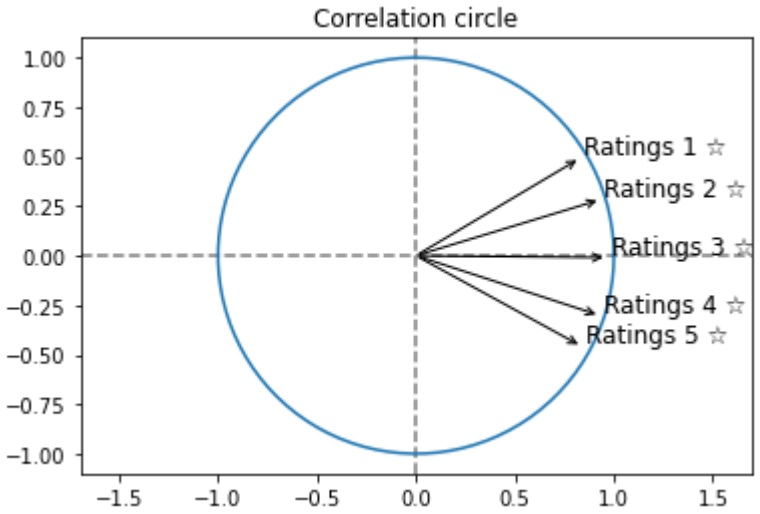
1. Afin de normaliser les données, on peut soustraire chaque colonne de la matrice de score par sa moyenne puis diviser par son écart type. On n'oublie pas aussi de transposer la matrice de score afin d'avoir les différents scores en attributs.
2. Pour obtenir la matrice de covariance, on utilise la fonction pandas "corr".
3. Finalement, pour calculer les valeurs propres et les vecteurs propres, on utilise la fonction numpy "linalg.eig". Cette fonction retourne dans l'ordre décroissant les valeurs propres et leurs vecteurs propres associés.

Voici maintenant la variance expliquée par chaque dimension :



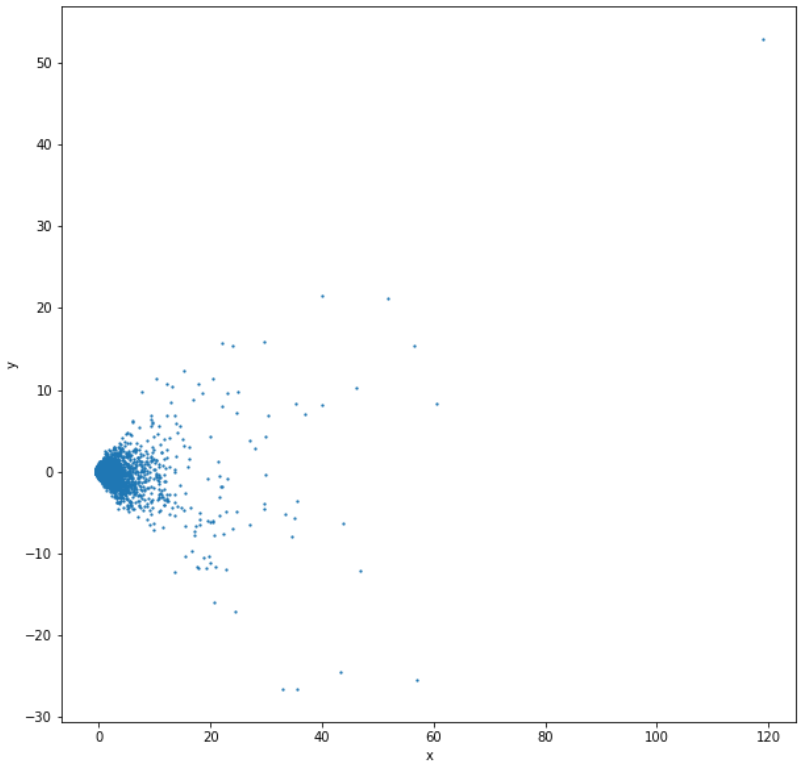
On remarque qu'en prenant deux dimensions, on arrive à expliquer plus de 90% de la variance de nos données ce qui est très bon car on veut représenter les points dans un plan (par conséquent on doit considérer les deux premières dimensions).

Vérifions maintenant comment nos attributs (i.e les différents : score = 1, score = 2, etc..) sont représentées par ces deux dimensions :

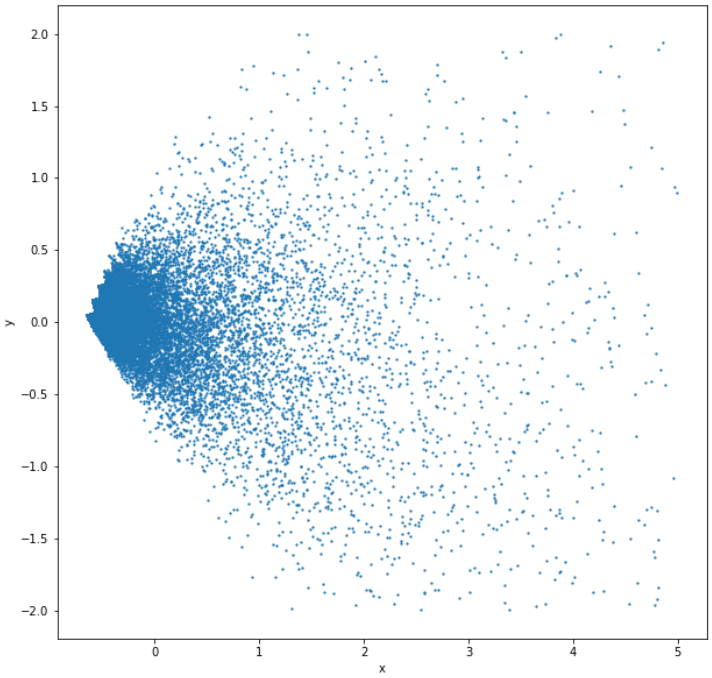


On remarque que tous les attributs sont très bien exprimés car chacune des flèches est très proche du cercle. Par ailleurs, on peut s'attendre à une distribution particulière des points : les livres les moins appréciés seront placés en hauteur du plan, les livres plus-ou-moins appréciés seront en milieu et les livres les plus appréciés seront en bas du plan. Afin de projeter les livres sur le plan, il faut maintenant multiplier la matrice de score par la matrice de taille 5x2 contenant les deux vecteurs propres associés aux deux premières valeurs propres les plus grandes (pour projeter sur le plan (dim1, dim2)). Pour cela, on utilise la fonction "dot" de pandas du dataframe contenant les données normalisées.

Voici maintenant les données projetées sur le plan :

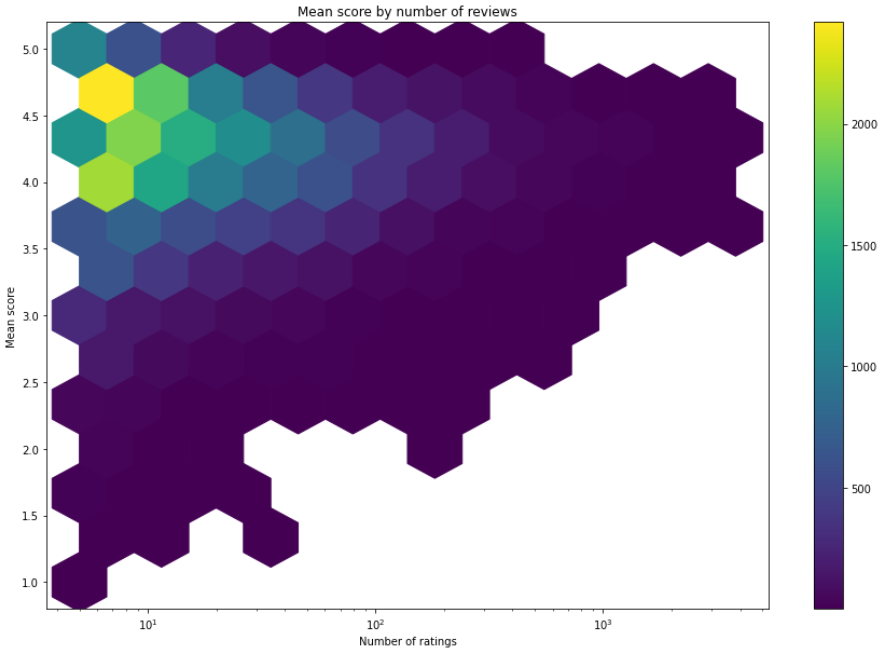


Afin d'obtenir un meilleur affichage, on peut supprimer les points "extrêmes" ou "outliers" :



b.2) Coloriage du plan selon l'alternative de la question a.5) et histogramme.

Avant d'implémenter la moyenne bayésienne, il est nécessaire de déterminer les paramètres C et m. Pour cela, nous allons afficher un graphe "hexbin" permettant de connaitre la tendance générale de notre jeu de données :

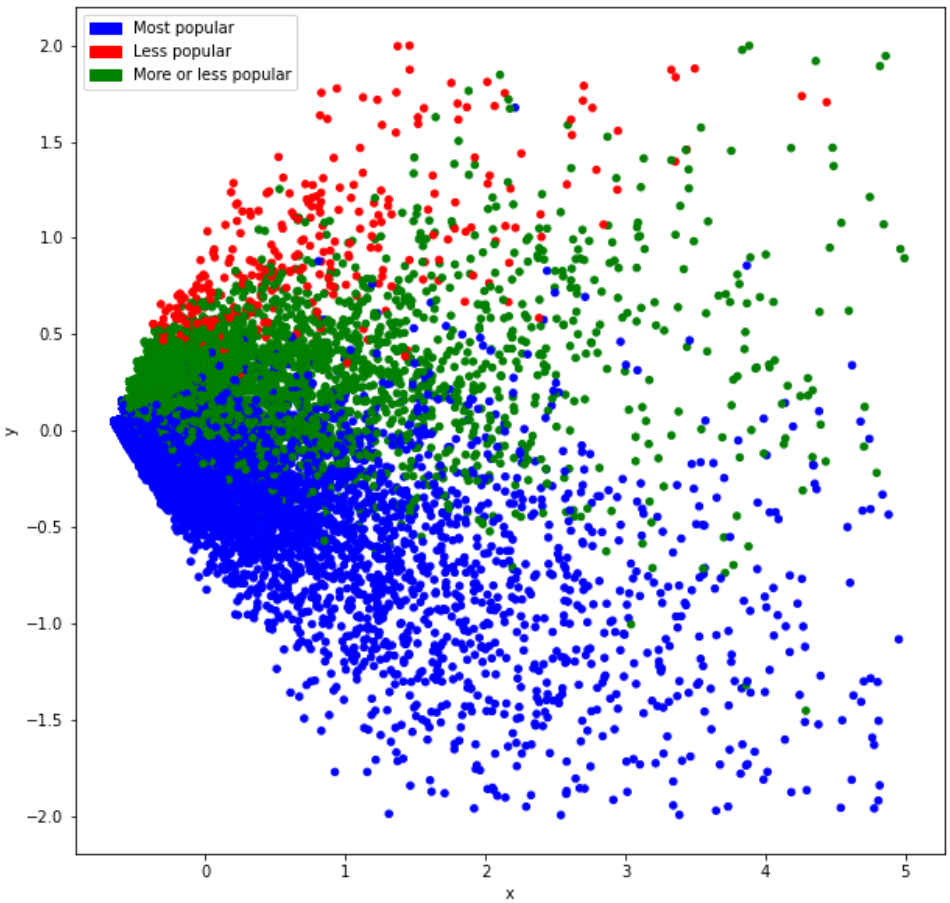


On remarque qu'il y a un cluster de données (en haut à gauche, voir le gradient de couleur jaune – vert) ayant très peu d'avis mais de très bons scores. Par conséquent, la tendance générale de notre jeu données est que la majorité des livres ont un très bon score tout en ayant très peu d'avis. Ainsi, on peut fixer C = 4.0 et m = 20, ce qui représente la frontière externe de ce cluster : cela signifie qu'on peut s'attendre à un score de 4 étoiles si on a environ 20 avis.

Pour calculer la moyenne bayésienne, on utilise la fonction getBayesianScore : celle-ci utilise la moyenne pondérée et les paramètres C, m. Voici le résultat obtenu trié dans l'ordre décroissant :

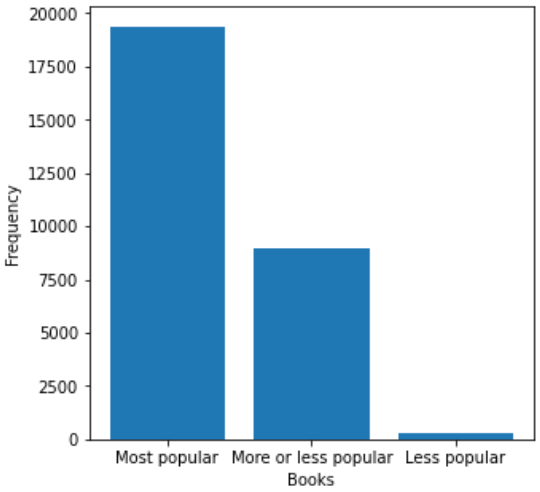
	0195067142	0061906220	0060256672	0062225227	0152056610	0060525614	0140502343	029764680X	0141345713
Bayesian score	4.846473	4.811429	4.804878	4.790323	4.781818	4.778846	4.778409	4.776758	4.766912
Number of ratings	462.000000	155.000000	185.000000	228.000000	90.000000	84.000000	156.000000	307.000000	1340.000000

On remarque que les meilleurs livres sont maintenant ceux avec beaucoup plus de reviews, ce qui est nettement mieux qu'avec la moyenne pondérée. Afin de colorier les différents points projetés, il est nécessaire fixer un intervalle dans lequel on estime que chaque livre tombe dans une catégorie : l'énoncé suggère [2.5, 3.5]. Cependant, comme nous l'avons vu, on a énormément de livres très bien notés : en effet, les gens ont tendance à noter favorablement lorsqu'ils donnent leurs avis. Par conséquent, on a changé d'intervalle par un beaucoup plus sévère : [3.5, 4.0]. Cela signifie pour qu'un livre soit apprécié il faut qu'il ait un score strictement supérieur à 4.0, si son score tombe dans l'intervalle alors il est moyennement apprécié et finalement si inférieur à 3.5 alors il n'est pas du tout apprécié. Voici le nuage colorié que nous obtenons :



On remarque directement les différents clusters, cependant leurs frontières ne sont pas nettes. Bien plus, comme nous l'a montré l'étude du cercle de corrélation, on retrouve le même agencement, i.e : les livres les plus appréciés en bas (bleue), les plus ou moins appréciés au milieu (vert) et finalement les moins appréciés en haut (rouge).

L'histogramme de cette distribution, est obtenu en comptant les points bleues, verts et rouges :



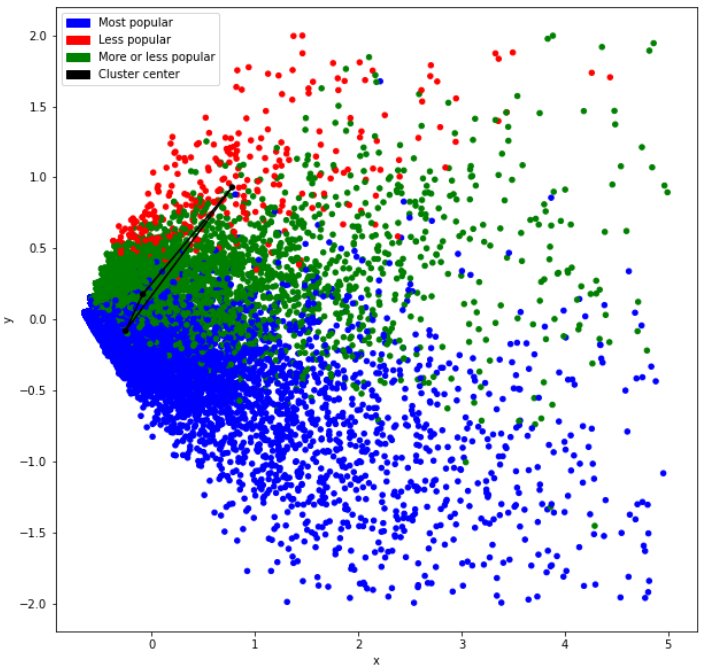
La fréquence d'apparition de chaque catégorie suit celle du coloriage, ce qui est attendu.

b.3) Centre de chaque cluster.

Afin de calculer le centre de chaque cluster, on somme les coordonnées de chaque point le contenant puis on divise par la cardinalité du cluster. En ayant trois clusters différents, on obtient trois centres :

```
blue cluster coordinates : (-0.25333116763695634,-0.07875410932765767)
red cluster coordinates : (0.7788944514461678,0.930929138190557)
green cluster coordinates : (-0.08630922355634271,0.17900556196284914)
```

Avec ces trois points, on peut former un triangle :



On remarque que chacun de ces centres ne représentent pas globalement tous les points. En effet, étant donné qu'on a une concentration des points au niveau de l'origine du plan, les centres vont se trouver près de cette origine. Cependant, chaque cluster s'étend horizontalement à droite à travers une grande dispersion de points (très prononcé au niveau du cluster bleue). Par conséquent, les centres ne représentent pas correctement la totalité des points.

c) Etude mensuelle

Nous cherchons à présent à savoir comment les tendances statistiques mesurant les opinions évoluent mensuellement à partir de notre dataset initial. Pour cela, nous commençons par filtrer nos données sur une année choisie arbitrairement (ici 2013) que l'on divise ensuite en 12 dataframes, soit un par mois.

Notre fonction extractByYear va ainsi récupérer, à partir du jeu de données initial, les livres dont la « reviewTime » contient l'année voulue. Nous convertissons ensuite l'attribut "unixReviewTime" en une date lisible à l'aide d'une fonction de pandas et nous pouvons alors créer notre dictionnaire composé des 12 data frames mensuels. Lors de l'extraction chaque dataframe est ensuite transformé en une matrice de score à l'aide de la fonction extractScore.

Nous pouvons, dès lors, recommencer les opérations statistiques effectuées précédemment mais cette fois ci sur les 12 nouveaux dataframes. Nous avons mis sous forme de fonctions la plupart des opérations à réutiliser, nous n'avons donc pas à les recoder mais simplement à gérer l'affichages des résultats obtenus mensuellement via ces fonctions. Pour cela, la méthodologie est la même dans les différentes étapes qui suivent : les nouvelles matrices sont obtenues en bouclant sur tous les éléments du dictionnaire contenant toutes les matrices de score par mois que l'on va fournir à la fonction permettant d'effectuer l'opération souhaitée.

Premièrement, nous affichons les dataframes contenant les moyennes pondérées pour chaque mois à l'aide de la fonction getAverageScore :

January

	006226768X	0151008116	0263899403	006230240X	0307346609	0007230206	0060175400	0061147958	0061231401	0061493341	...	0060935618	0099283212
Mean score	4.05	4.257143	4.167203	4.228495	4.065574	3.588235	4.12	3.857143	4.236842	3.578947	...	5.0	5.0
Number of ratings	80.00	210.000000	311.000000	372.000000	61.000000	51.000000	25.00	14.000000	38.000000	19.000000	...	1.0	1.0

Exemple affichage moyenne pondérée pour janvier 2013

Deuxièmement, nous trions les livres dans l'ordres croissant selon la moyenne de leurs notes afin de pouvoir ensuite afficher les meilleurs, avec la fonction getBestBooks, et les mauvais livres, avec la fonction getWorstBooks :

May

	0062700847	0307475395	0142000329
Mean score	5.0	5.0	5.0
Number of ratings	1.0	1.0	1.0

Affichage 3 meilleurs livres en mai 2013

May

	0060014164	006000925X	0099324210
Mean score	1.0	1.0	1.0
Number of ratings	1.0	1.0	1.0

Affichage 3 pires livres en mai 2013

Troisièmement, nous affichons le premier quart des livres les plus appréciés pour les 12 mois, grâce à la fonction getFirstQuarterBestBooks :

December

	0062503731	0060090383	0142427543	0142427489	0142427225	0060088877	0142426059	0142424722	0060087714	0142500046	...	0140434925	0140194967
Mean score	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	...	5.0	5.0
Number of ratings	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0	1.0

2 rows × 1778 columns

Affichage premier quart des livres les plus appréciés en décembre 2013

Enfin, afin de montrer l'évolution des différents clusters mensuellement durant une année fixée, on va utiliser un GIF. L'idée est de construire un GIF à partir de chaque image mensuelle présentant une projection pour ce dit mois.

Pour chaque mois, on va projeter sa matrice de score normalisée et transposée dans le même espace vectoriel trouvé à la Section (b). Comme métrique de classification, on va utiliser la moyenne bayésienne : pour fixer de manière automatique le paramètre C, on va se baser sur le premier quantile du nombre de reviews du mois et concernant le m, on va calculer la moyenne des scores de tous les livres de ce mois aussi. En résumé, cela revient à faire pour chaque mois "i":

1. Calculer sa matrice de score moyenne
2. Obtenir les paramètres C et m
3. Calculer sa matrice de moyenne bayésienne à l'aide du point "1."
4. Projeter sa matrice de score dans l'espace obtenu à la question b)

En ayant maintenant toutes ces données, on peut facilement obtenir le GIF souhaité à l'aide de la fonction createGIF : celle-ci enregistre chaque image mensuelle en une "frame" puis à l'aide du paramètre "ifDelete" peut supprimer les images générées si l'on souhaite obtenir que le GIF. Finalement, on utilise aussi la fonction getAxesInterval, permettant d'obtenir les [xmin, ymin] et [xmax, ymax] de tous les points afin d'obtenir les mêmes axes sur chaque image.

Le code couleur utilisé pour ces nouvelles projections est le même que celui utilisé à la question b). Cependant, nous avons légèrement changé l'intervalle de satisfaction à [3.0, 4.0]. C'est à dire qu'un livre apprécié aura un score strictement supérieur à 4.0, un livre moyennement apprécié sera entre 3.0 et 4.0 et un livre pas du tout apprécié sera noté moins de 3.0.

Ainsi, pour l'année 2013 nous obtenons la génération de ce GIF : lien . Dans le cas inverse, vous trouverez le GIF aussi dans le notebook ou bien à la racine du dossier du TP.

Interprétation :

De nouveau, nous pouvons aisément distinguer les différents clusters, mais leurs frontières ne sont toujours pas nettes. L'agencement des points reste cohérent par rapport à la projection dans le plan de la question b) c'est-à-dire que l'on retrouve les livres les plus appréciés en bas (bleue), les plus ou moins appréciés au milieu (vert) et les moins appréciés en haut (rouge).

Cependant, ce GIF à l'avantage de nous permettre de visualiser facilement les différences de notations selon les mois. En effet, on remarque sur certains mois, comme celui d'octobre, qu'il y a eu moins de revue car il y a moins de points apparants. Enfin, les couleurs du nuage nous permettent quant à elles de nous rendre compte des tendances de notations selon les mois de l'années, malgré le fait que la concentration de points par couleur semble relativement équitable tout au long de l'année.