UNIVERSITÉ DE NANTES

Rapport de projet M1 AD 1920

Aniss Bentebib, Camille-Amaury Juge

Respectivement M1 Informatique ATAL et DS

Sommaire :

I - Introduction

Ce rapport a été réalisé dans le cadre du projet de M1 informatique à l'université de Nantes, pour le cours d'Analyse de données tenu par Mme Lise Bellanger-Husi.

Nous allons donc commencer par présenter brièvement le jeu de données ainsi que les différents enjeux de ce projet.

Tout d'abord, il faut mentionner la taille du jeu de données : celui-ci est de 40 individus par 1000 variables. Les individus représentent des patients sains et d'autres atteints d'une maladie non mentionnée.
On sait d'ores et déjà que les 20 premiers patients sont sains et les 20 derniers sont malades.

Quant aux variables, elles représentent des gènes qui ont été prélevé sur ces fameux patients. L'unité est identique pour toutes les mesures ce qui facilite grandement l'exploitation. Ce sont des variables quantitatives.

De plus, on remarque qu'il y a des valeurs positives et négatives. Pour conclure, il ne manque pas de données.

Le but est donc le suivant : peut-on mettre en évidence des gènes qui seraient identifiables comme étant la cause de cette maladie chez les patients ?

Pour cela, nous avons décidé d'opérer en plusieurs étapes, nous vous présenterons dans un premier temps les implémentations et les fonctionnalités de notre code puis nous poursuivrons sur les exploitations du jeu de données.

Dans un premier temps, prenons le temps d'expliquer la structure du projet. Le langage choisis est R.
Ce dernier a été réalisé au moyen de l'outil Git/Github afin de travailler en collaboration, mais aussi de faciliter la documentation du projet :

https://github.com/camilleAmaury/R_data_analysis_project

Un fichier Readme.md est disponible afin de donner toutes les instructions nécessaires à l'utilisation du module.

De plus, nous avons créé un dossier test, où nous avons comparé toutes nos fonctions aux fonctions implémentées dans différents module en R. Il suffit simplement d'exécuter les fichiers et tout l'affichage détaillera les résultats.

Finalement, tout le projet a été documenté et réalisé en anglais.

II - Code et Implémentation

Ici, nous détaillerons l'implémentation et les choix de nos différentes fonctions.

A - ACP

Dans le cadre de l'analyse multidimensionnelle que nous devons mener, notre choix s'est porté sur l'Analyse en Composant Principale.

Étant donné que les variables sont quantitatives, l'ACP est un choix judicieux. Celle-ci permet aussi de réduire le nombre de dimension en préservant une part d'information raisonnable afin de faciliter la compréhension et l'interprétation par l'humain.

Penchons-nous sur notre implémentation :

Dirigez-vous vers le fichier du projet : "***R/pca.R***"

La fonction "***pcafunction***" se divise en 3 sous-partie que nous allons détailler ci-dessous.

Avant toute chose, nous allons évoquer les paramètres :

- ***M*** : DataFrame ou Matrix, la matrice initiale des individus et variables
- ***center*** : booléen, Si vous souhaitez centrer les données de la matrice M (vrai par défaut)
- ***scale*** : booléen, Si vous souhaitez centrer et réduire les données de la matrice M. Attention scale écrase la variable center et la met forcément à vraie (vrai par défaut).
- ***bias*** : booléen, Si vous souhaitez réaliser une acp non biaisée (false) ou biaisée (true). Ce paramètre influe sur la matrice de covariance.
- ***Q*** : Matrix, matrice représentant la métrique des variables (par défaut diag(1/p))
- ***D*** : Matrix, matrice représentant la métrique des individus (par défaut diag(1/n))
- ***axisMethod*** : string, méthode utilisée si l'on souhaite déterminer automatiquement le nombre d'axe à conserver : "*elbow*" pour la méthode du coude, "*kaiser*" pour la méthode de kaiser.
- ***axisNumber*** : Integer, choix de l'utilisateur concernant le nombre d'axe à garder : (par défaut c'est ***axisMethod*** qui détermine le nombre d'axe, sauf si vous précisez un nombre >= 2)

1 - Éléments de base

Pour accéder au retour de ce composant : "***votrevariable$base_component$paramètre***"

Ce module comprend comme son nom l'indique le pré-traitement des données dans le but de réaliser l'ACP.

Voici le retour de cette fonction avec les différentes variantes dues au paramètres :

- ***base*** : Matrix, matrice initiale
- ***rownames*** : vector[string], les noms des lignes du dataframe ou i1 à in
- ***colnames*** : vector[string], les noms des colonnes du dataframe ou v1 à vp
- ***G*** : Matrix, matrice représentant le centre de gravité de chaque variable (uniquement si ***scale*** ou ***center*** sont à TRUE)
- ***M_centered*** : Matrix, matrice centrée des données initiales (uniquement si ***scale*** ou ***center*** sont à TRUE)
- ***S*** : Matrix : matrice de covariance des données initiales (uniquement si ***scale*** ou ***center*** sont à TRUE). Si ***bias*** est à TRUE, la matrice est biaisée.
- ***M_scale*** : Matrix, matrice des données centrées et réduites (uniquement si ***scale*** est à TRUE).
- ***D1_div_sd*** : Matrix, matrice diagonale comportant 1/écart-type des variables initiales (permettant le calcul de la matrice de corrélation) (uniquement si ***scale*** est à TRUE).
- ***R*** : Matrix, matrice de corrélation des données initiales (uniquement si ***scale*** est à TRUE).

Tous ces retours ont été testé dans le fichier "***R/test/test_pca.R***". Veuillez-vous y référer pour comprendre l'utilisation plus en détails.

2 - Éléments de l'ACP

Pour accéder au retour de ce composant : "***votrevariable$PCA_component$paramètre***"

Ce module comprend tous les éléments de l'ACP.

Voici le retour de cette fonction avec les différentes variantes dues au paramètres :

- ***Si*** : Matrix, matrice composée par le triplet X Q D de l'acp
- ***X*** : Matrix, matrice correspondant au X du triplet de l'acp (en fonction des paramètres c'est soit ***M***, soit ***M_centered*** soit ***M_scale***)
- ***values*** : vector[Real], les valeurs propres hormis celles égales à 0
- ***vectors*** : Matrix, les vecteurs propres en fonction du rang de la matrice ***Si*** (axes principaux)
- ***rank*** : Integer, le rang de la matrice ***Si***
- ***axisNumber*** : Interger, le nombre d'axes conservés
- ***Fi*** : Matrix, Composante principales, matrice représentant la projection des individus sur les différents axes
- ***Gi*** : Matrix, matrice représentant la projection des variables sur les différents axes
- ***Inertia*** : Real, Inertie de ***Si*** (somme des valeurs propres)
- ***axisIC*** : vector[Real], pourcentage d'information capté par les axes
- ***cumulative_axisIC*** : vector[Real], pourcentage d'information cumulé capté par les axes

Tous ces retours ont été testé dans le fichier "***R/test/test_pca.R***". Veuillez-vous y référer pour comprendre l'utilisation plus en détails.

3 - Éléments d'interprétation

Pour accéder au retour de ce composant : "***votrevariable$PCA_interpretation$paramètre***"

Ce module comprend tous les éléments permettant d'interpréter les résultats de l'ACP.

Voici le retour de cette fonction avec les différentes variantes dues au paramètres :

- ***qlt_Fi*** : Matrix, matrice représentant la qualité de représentation des individus sur le nombre d'axes conservés : ***axisNumber***
- ***ctr_Fi*** : Matrix, matrice représentant la contribution absolue des individus sur les axes conservés.
- ***qlt_Gi*** : Matrix, matrice représentant la qualité de représentation des variables sur le nombre d'axes conservés : ***axisNumber***
- ***ctr_Gi*** : Matrix, matrice représentant la contribution absolue des variables sur les axes conservés.

Tous ces retours ont été testé dans le fichier "***R/test/test_pca.R***". Veuillez-vous y référer pour comprendre l'utilisation plus en détails.

Pour conclure sur l'ACP, si nous comparons à l'existant à savoir "***dudi.pca***", notre fonction se veut moins généraliste que cette fonction (nous ne sommes pas sûrs de pouvoir réaliser une ACP avec une matrice M non centrée et non réduite) en revanche nous prenons en charge l'ACP normée et non-normée, biaisée ou non.

De plus, notre fonction fournit tous les éléments de calculs sans exception, ce qui permet d'avoir un regard global sur l'ACP et de réutiliser chaque particule si besoin. On notera que "dudi.pca" ne retourne pas la plupart des matrices puisqu'il est capable de les recalculer avec les paramètres qu'ils retournent.

B - Kmeans

Afin de classer les individus sous forme de clusters, après avoir réalisé l'ACP, nous avons décidé d'implémenter k means, l'algorithme de clustering en apprentissage non-supervisé.

Penchons-nous sur notre implémentation :

Dirigez-vous vers le fichier du projet : "***R/Kmeans.R***"

La fonction "***kmeansfunction***" se divise en 3 sous-partie que nous allons détailler ci-dessous.

Avant toute chose, nous allons évoquer les paramètres :

- ***M*** : DataFrame ou Matrix, la matrice initiale des individus et variables
- ***k*** : Integer, Le nombre de cluster souhaité (par défaut si k est égal à -1 alors l'algorithme fera toutes les itérations de 2 à **kMax** pour sélectionner le meilleur k)
- ***initialization*** : string, La méthode utilisée pour choisir les k premiers centres (par défaut "***random***" : tirage aléatoire des k centres, "***kmeans++***" premier centre choisi aléatoirement puis le reste est déterminé par la distance euclidienne au point précédent, cette méthode est censée être plus efficace)
- ***d*** : string, La distance utilisée dans l'itération des k-clusters (par défaut "***euclidian***" distance euclidienne, "***mahalanobis***" qui prend en compte la corrélation et la variance entre des données et peut s'avérer plus efficace)
- ***precision*** : Real, La précision avec laquelle on veut que l'algorithme s'arrête : c'est à dire si le changement des distances entre l'étape n et n+1 des itérations et inférieur à ***precision*** alors arrêt.
- ***logs*** : booléen, Fut utile pour vérifier le déroulement de l'algorithme, peut aider à sa compréhension.
- ***seed*** : Integer, seed permettant de choisir un pattern dans l'aléatoire pour effectuer des tests répétables.
- ***kMax*** : Integer, choix de l'utilisateur concernant le nombre max de clusters (il doit forcément être inférieur à n individus et supérieur à 2) (par défaut, kMax sera égal à Min(10, individus-1)

1 - Choix initial des clusters

Dans une volonté de rendre notre fonction polyvalente, nous avons convenu d'implémenter plusieurs initialisation des k-clusters.

Pour cela, nous avons choisi la méthode classique par défaut de choix aléatoire sans remise. En effet, on choisis les k-centres comme étant des points appartenant au données initiales de manière aléatoire.

Un des problèmes de cette méthode est qu'elle peut choisir des centres très proche qui vont se concurrencer, alors qu'avec un choix plus judicieux de clusters ces deux points seraient sûrement réunis sous le même cluster.

C'est pour cela que nous avons choisi de rajouter l'algorithme kmeans++, qui sans détailler la preuve ici, retourne une approximation d'une initialisation des k-centres optimale.

Cet algorithme sélectionne aléatoirement un premier point des données initiales puis avec une distance choisie (ici nous n'avons implémenté que la distance euclidienne) donne une probabilité à chaque point d'être le prochain centre en fonction de sa distance.
Ainsi cela augmente la probabilité d'avoir des clusters plus éloignés et donc plus proche d'un résultat qualitatif.

2 - Distances

Pour poursuivre dans cette volonté de polyvalence de l'algorithme kmeans, nous avons décidé d'implémenter plusieurs distance qui interviennent au moment de choisir l'appartenance de chaque point aux différents centres.

Nous avons gardé la traditionnelle distance euclidienne qui est généralement la plus utilisée avec cette algorithme.

Nous avons pensé qu'il était bon d'implémenter une autre distance, qui collerait plus avec l'interprétation de l'ACP et du cas du clustering. En effet, la distance de Mahalanobis a l'avantage de prendre en compte les corrélations entre les différents paramètres et leur variance. Dans un sens, on pourrait penser que cette distance s'avère plus efficace pour ce genre de méthode.

3 - Choix par force brute

Comme indiqué plus ci-dessus, le choix des k clusters se réalise au moyen de deux paramètres : ***k*** et ***kMax***.

Nous avons implémenté la fonction de telle sorte que si k ne soit pas précisé par l'utilisateur, que celle-ci détermine automatiquement le k jugé le plus pertinent.

Pour cela, nous avons procédé avec la fonction "***selectBestKvalue***" qui est interne à la fonction "***kmeansfunction***".

Cette fonction effectue pour chaque itération de 2 à **kMax** l'algorithme kmeans, puis stocke les résultats dans une matrice qui est ensuite donnée à un algorithme de comparaison.

Cette algorithme de comparaison n'est autre que "***silhouette_imp***" qui comme son nom l'indique compare les différents résultats par la méthode des silhouettes. Cette méthode à l'avantage de définir chaque point en fonction de la qualité de sa représentation dans son propre cluster mais aussi dans tous les clusters voisins.

La fonction retournera le **bestK** choisi afin de donner à l'utilisateur un moyen de réutiliser manuellement le **bestK** après une première exécution.

Sinon, l'utilisateur peut préciser directement le ***k*** qu'il désire effectuer. Ainsi l'algorithme n'itère que pour un nombre de clusters donnés.

Avant de poursuivre, nous voulions notifier que notre but en implémentant kmeans était de pallier aux désavantages évoqués sur la page Wikipédia Kmeans de la sous-partie "*Avantages et inconvénients pour l'apprentissage*".

C - Plot And Explain

Dans un objectif d'interprétation des fonctions que nous avons implémenter précédemment

1 - ACP

Dans le but d'interpréter nos résultats, nous avons créé une fonction qui automatise la lecture des résultats de l'ACP.

Penchons-nous sur notre implémentation :

Dirigez-vous vers le fichier du projet : "***R/plotAndExplain.R***"

Portez votre attention sur la fonction "***explainPCA***"

Avant toute chose, nous allons évoquer les paramètres :

- ***acp*** : List of List, retour de notre fonction ACP.
- ***individualRate*** : Real, Le taux grâce auquel les individus sont considérés comme contribution majeure à un des axes (par défaut 1/n)
- ***variableRate*** : Real, Le taux grâce auquel les variables sont considérées comme contribution majeure à un des axes (par défaut 1/p)

Ainsi, la fonction va afficher la contribution positive et négative des individus/variables qui contribuent majoritairement à chaque axe en les listant.

De plus, nous avons affiché le cercle des corrélations lorsque le nombre d'axe retenus est de 2, puis les projections des individus sur les axes.

2 - Analyse monodimensionnelle

Afin de répondre au sujet, nous avons réimplémenter de manière plus détaillée la fonction "***summary***" de R.

Penchons-nous sur notre implémentation :

Dirigez-vous vers le fichier du projet : "***R/plotAndExplain.R***"

Portez votre attention sur la fonction "***explainDataset***"

Avant toute chose, nous allons évoquer les paramètres :

- ***M*** : List, données initiales.
- ***colnames*** : vector[string], vecteur comprenant le nom des variables
- ***bidimensionnal*** : booléen, Si l'analyse a effectuer est bivariée ou univariée

Pour l'analyse Univariée :

la fonction affiche trois catégories :

- *paramètres de position :* (moyenne, médiane, mode) et l'interprétation de l'écart entre la moyenne et la médiane avec un taux choisi de 25% (si supérieur à 25% de différence alors cela veut dire qu'il y a potentiellement des valeurs extrêmes qui tirent la moyenne) —> d'où l'intérêt de centrer et réduire.

- *paramètres de dispersion :* (écart-type, étendue, écart interquartile) : affichage de ses différents paramètres, plus ils sont élevés plus ceux-ci indiquent une grande dispersion des valeurs. (attention à l'étendue qui n'est pas toujours représentative)

- *paramètres de forme* : (kurtosis, skewness) : affichage des taux par les fonctions de R, et l'interprétation textuelle.

Pour chaque variable, on affiche un diagramme à moustache mais aussi une représentation de la répartition par un histogramme.

3 - Analyse bidimensionnelle

La fonction est identique à celle citée ci-dessus, le paramètre "***bidimensionnal***" doit être spécifié à TRUE

Pour l'analyse Bivariée :

la fonction affiche trois catégories :

- *Test du Chi-deux d'indépendance :* (X_square, df, p-value) si p-value est plus petit que le seuil 0.05 alors les variables comparée ne sont pas indépendantes et inversement. X-square évolue de manière inverse, plus il est grand plus les variables sont dépendantes.

- *Régression Linéaire :* affichage de ses différents paramètres, plus ils sont élevés plus ceux-ci indiquent une grande dispersion des valeurs. (attention à l'étendue qui n'est pas toujours représentative)

4 - Kmeans

En complément, nous avons affiché les résultats de la fonction Kmeans.

Penchons-nous sur notre implémentation :

Dirigez-vous vers le fichier du projet : "***R/plotAndExplain.R***"

Portez votre attention sur la fonction “***explainKmeans***”
Avant toute chose, nous allons évoquer les paramètres :

- ***M*** : Matrix, matrice des projections des individus à classifier.
- ***kmeans***: List, retour de notre fonction kmeans

Ainsi, nous avons décidé de représenter sous la forme d’un plan et de points colorés en fonction de l’appartenance au cluster.

D - QR decomposition, valeurs et vecteurs propres.

En guise de bonus, nous avons tenté de ré-implémenter l’algorithme qui permet de trouver les valeurs propres et vecteurs propres d’une matrice.

Il existe de nombreux algorithmes avec des méthodes qui diffèrent pour le réaliser, nous avons choisi l’algorithme QR avec la variante des matrice de HouseHolder.

Cet algorithme a l’avantage d’être numériquement stable (peu de division en nombre flottant) et en complexité polynomiale.

Penchons-nous sur notre implémentation :

Dirigez-vous vers le fichier du projet : “***R/eigenfunction.R***”

Portez votre attention sur la fonction “***eigen_function***”

la fonction est similaire à la fonction “***eigen***” en R, c’est à dire qu’elle retourne sous le même format.

En revanche, dans nos test nous nous sommes rendus compte qu’il y a avait de plus en plus d’incohérence au fur et à mesure des dimensions de la matrice, mais aussi des résultats erronés quant aux vecteurs propres lorsque la matrice initiale n’est pas symétrique.

Nous n’avons malheureusement pas pu interpréter cette erreur en dehors du fait des approximations numéraires et des erreurs d’arrondis.

III - Interprétation du jeu de données

Dans cette partie, nous évoquerons le résultat des analyses que nous avons menés sur le jeu de données.

A - Analyse Univariée des variables

Voir annexes.

B - Analyse Bivariée des variables

Voir annexes.

C - Analyse Multidimensionnelle des variables

Voir annexes.

Tout a été automatisé dans le but de faciliter l'interprétation.

D - Clusterisation des individus

Veuillez vous référer au fichier "***result/kmeans.html***".

Comme nous pouvons le remarquer, la clusterisation des composantes principales de l'ACP nous montre clairement qu'il y a deux clusters.

Les individus 1 à 20 sont dans le premier cluster et les individus de 21 à 40 dans le second.
Ceci nous indique que Kmeans a automatiquement détecté et clusterisé les patients malades et sains si l'on se réfère à la véracité de l'énoncé.

IV - Conclusion

Malgré le manque d'information sur le type de données que nous possédons (l'unité, quelles mesures sont effectuées sur les différents gènes et que représentent-elles, la maladie en question) on peut tout de même isoler le premier axe qui semble correspondre à la segmentation des gènes et des individus en fonction de la maladie et de la résistance à cette maladie.

On nuancera tout de même les propos puisqu'avec seulement deux axes, et un pourcentage très faible, les variables sont tout de même mal représentées sur le cercle de corrélation. Ainsi la qualité des résultats est à prendre avec des pincettes.

Idéalement, il faudrait un taux d'informations sur les axes plus ou moins supérieurs à 70% pour avoir un résultat qui se rapproche vraiment de la réalité, mais dans ce jeu de données, cela reviendrait à considérer au moins ¾ des axes, ce qui réduit légèrement le problème mais ne permet pas une interprétation graphique humaine efficace.

V - Sources

Principale source de documentation et d'implémentation des fonctionnalités :

QR decomposition : https://en.wikipedia.org/wiki/QR_decomposition?oldid=378686082
http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf
Matrice de HouseHolder : https://en.wikipedia.org/wiki/Householder_transformation
http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf
Silhouette : https://en.wikipedia.org/wiki/Silhouette_(clustering)
Kmeans : https://en.wikipedia.org/wiki/K-means_clustering
Kmeans++ : https://en.wikipedia.org/wiki/K-means%2B%2B
http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf (très intéressant au passage).
Mahalanobis : https://en.wikipedia.org/wiki/Mahalanobis_distance
https://blogs.sas.com/content/iml/2012/02/15/what-is-mahalanobis-distance.html
Cours de M1 informatique dispensés par Mme Lise Bellanger-Husi.

Package R utilisés, comparés :

dudi.pca : https://pbil.univ-lyon1.fr/ADE-4/ade4-html/dudi.pca.html
inertia.dudi :
https://www.rdocumentation.org/packages/ade4/versions/1.7-13/topics/inertia.dudi
e1071 : https://cran.r-project.org/web/packages/e1071/index.html