X1MS010 - Rapport de projet

Analyse des données

Lucas LELIEVRE — M1 Informatique VICO

Thomas LAPIERRE — M1 Informatique ATAL

Description des données

Analyse générale du jeu de données

Tout d'abord, nous avons commencé par importer les données afin de pouvoir les comprendre et ainsi mieux comprendre le projet. A première vue, on a pu remarquer que de nombreuses données étaient manquantes ou représentées par des tirets. On a conclu qu'une phase de traitement de ces données serait indispensable. A deuxième vue, on a pu remarquer que les colonnes n'étaient pas au bon format. En effet, les colonnes possédant les données quantitatives étaient au format "*character*" au lieu de "*numeric*".

Dans un premier temps, nous avons converti les colonnes contenant les valeurs quantitatives de chaîne de caractère en valeur numérique. Dans un second temps, nous avons décidé d'isoler les données essentielles au projet. Autrement dit, de seulement garder les données concernant le fromage. Suite à ces étapes, le *dataframe* de départ contenait à présent 134 lignes et 76 colonnes contre 3186 lignes au départ.

Pour mener à bien ce projet, nous avons ensuite dû trouver le juste milieu de suppression de colonnes et de lignes afin d'éliminer toutes les valeurs "NA". Pour se faire, delladata nous a été d'une grande aide. Sur le site se présentent différentes fonctions afin de pouvoir visualiser et explorer les différentes données manquantes de notre jeu de données.
Nous avons tout d'abord commencé par utiliser la fonction "summary" afin d'avoir une vision globale sur nos données. Ensuite, nous avons cherché le nombre de données manquantes au total et par colonne (voir **Figure A**). Puis nous avons souhaité avoir des informations plus détaillées (représentation graphique, pourcentage) sur les données manquantes. C'est pourquoi nous avons, par le biais de librairies, établi une analyse plus profonde. La librairie *naniar* s'est révélée très utile parce qu'elle traduisait les données numériques de *funModeling* sous forme de graphiques. Sur la **Figure B** représentant la sortie de la fonction "df_status", on a principalement regardé les colonnes "p_zeros" et "p_na". "p_zeros" nous a permis d'éliminer l'alcool. En effet, le pourcentage de 0 pour cette colonne étant de 100%, il allait être non intéressant d'utiliser cette variable pour l'ACP. Enfin, sur la **Figure C**, vous pourrez voir le graphique généré par *naniar* représentant le pourcentage de données manquantes par colonne. Nous avons décidé d'éliminer d'office, les variables pour lesquelles il y avait plus de 25% de NA. Suite à ce premier balayage de colonne, nous avons effectué des recherches sur internet pour déterminer les composants nutritionnels principaux. Autrement dit, le but était de trouver les variables les plus représentatives d'un fromage. Nous avons multiplié les sources afin de définir les variables suivantes comme fondamentales à la description d'un fromage : protéines, lipides, glucides, calcium, sodium, acides gras saturés. A donc suivi une phase de balayage par ligne en éliminant toutes les lignes ne possédant pas une des variables citées précédemment. Enfin, nous avons refait une analyse des NA par colonne, certaines variables possédaient toujours trop de NA, nous avons donc décidé de les supprimer. En revanche, pour les variables possédant moins de 10 NA par colonne, nous avons procédé à une suppression par ligne intelligente. Autrement dit, nous avons cherché à supprimer les lignes qui diminueraient le plus de NA par colonne différente. Nous avons terminé par devoir faire un choix entre soit garder le sucre soit l'amidon car garder les deux auraient fait supprimer trop de lignes et supprimer les deux

aurait rendu l'ACP moins intéressante. Nous avons décidé de garder la variable du sucre et de supprimer celle de l'amidon.

A la fin de ce processus, le dataframe initial devient un jeu de données composé de 20 variables et 89 lignes. Ainsi, 45 lignes ont été supprimées et 47 variables. Le détail du code de ce processus peut être consulté sur la **Figure D** de l'annexe.

Présentation unidimensionnelle

Energie, Règlement UE N° 1169/2011 (kJ/100 g)	Energie, Règlement UE N° 1169/2011 (kcal/100 g)	Energie, N x facteur Jones, avec fibres (kJ/100 g)	Energie, N x facteur Jones, avec fibres (kcal/100 g)
Min. : 630	Min. :151.0	Min. : 635	Min. :152.0
1st Qu.:1170	1st Qu.:281.0	1st Qu.:1170	1st Qu.:283.0
Median :1350	Median :325.0	Median :1360	Median :327.0
Mean :1332	Mean :321.1	Mean :1339	Mean :322.8
3rd Qu.:1510	3rd Qu.:364.0	3rd Qu.:1520	3rd Qu.:366.0
Max. :1900	Max. :457.0	Max. :1900	Max. :460.0
Eau (g/100 g)	**Protéines, N x facteur de Jones (g/100 g)**	**Protéines, N x 6.25 (g/100 g)**	**Glucides (g/100 g)**
Min. :28.70	Min. : 0.50	Min. : 0.50	Min. : 0.0000
1st Qu.:41.60	1st Qu.:17.90	1st Qu.:17.50	1st Qu.: 0.0000
Median :49.40	Median :21.20	Median :20.70	Median : 0.0000
Mean :48.09	Mean :20.84	Mean :20.44	Mean : 0.7096
3rd Qu.:54.10	3rd Qu.:24.20	3rd Qu.:23.80	3rd Qu.: 0.0000
Max. :72.50	Max. :34.00	Max. :33.30	Max. :21.2000
Lipides (g/100 g)	**Sucres (g/100 g)**	**Fibres alimentaires (g/100 g)**	**Polyols totaux (g/100 g)**
Min. : 9.30	Min. :0.0000	Min. :0.00000	Min. :0.00000
1st Qu.:23.30	1st Qu.:0.0000	1st Qu.:0.00000	1st Qu.:0.00000
Median :26.40	Median :0.0000	Median :0.00000	Median :0.00000
Mean :26.15	Mean :0.2406	Mean :0.06067	Mean :0.01124
3rd Qu.:30.00	3rd Qu.:0.2000	3rd Qu.:0.00000	3rd Qu.:0.00000
Max. :38.40	Max. :4.9500	Max. :2.60000	Max. :1.00000
Cendres (g/100 g)	**Acides organiques (g/100 g)**	**AG saturés (g/100 g)**	**AG monoinsaturés (g/100 g)**
Min. :1.220	Min. :0.0000	Min. : 1.87	Min. : 1.450
1st Qu.:2.500	1st Qu.:0.0000	1st Qu.:16.20	1st Qu.: 4.900
Median :3.110	Median :0.1200	Median :18.00	Median : 6.110
Mean :3.217	Mean :0.3744	Mean :17.23	Mean : 6.287
3rd Qu.:3.840	3rd Qu.:0.6800	3rd Qu.:19.50	3rd Qu.: 7.390
Max. :5.400	Max. :1.8800	Max. :25.70	Max. :19.500
AG polyinsaturés (g/100 g)	**Sel chlorure de sodium (g/100 g)**	**Calcium (mg/100 g)**	**Sodium (mg/100 g)**
Min. :0.1500	Min. :0.49	Min. : 32	Min. : 197.0
1st Qu.:0.5500	1st Qu.:1.28	1st Qu.: 180	1st Qu.: 505.0
Median :0.7400	Median :1.59	Median : 540	Median : 628.0
Mean :0.9082	Mean :1.65	Mean : 496	Mean : 648.9
3rd Qu.:0.9100	3rd Qu.:1.90	3rd Qu.: 722	3rd Qu.: 768.0
Max. :8.3800	Max. :3.68	Max. :1090	Max. :1470.0

Le tableau ci-dessus représente les données obtenues après avoir exécuté la commande "**summary**" sur notre jeu de données final.

Les variables Glucides, Sucres, Fibres alimentaires, Polyols totaux et acides organiques présentent un écart important dans leurs valeurs entre les individus. On peut supposer que beaucoup des individus ont ces variables proches ou égales à zéro, tandis qu'un petit nombre d'individus auront ces variables à une valeur plus élevée.

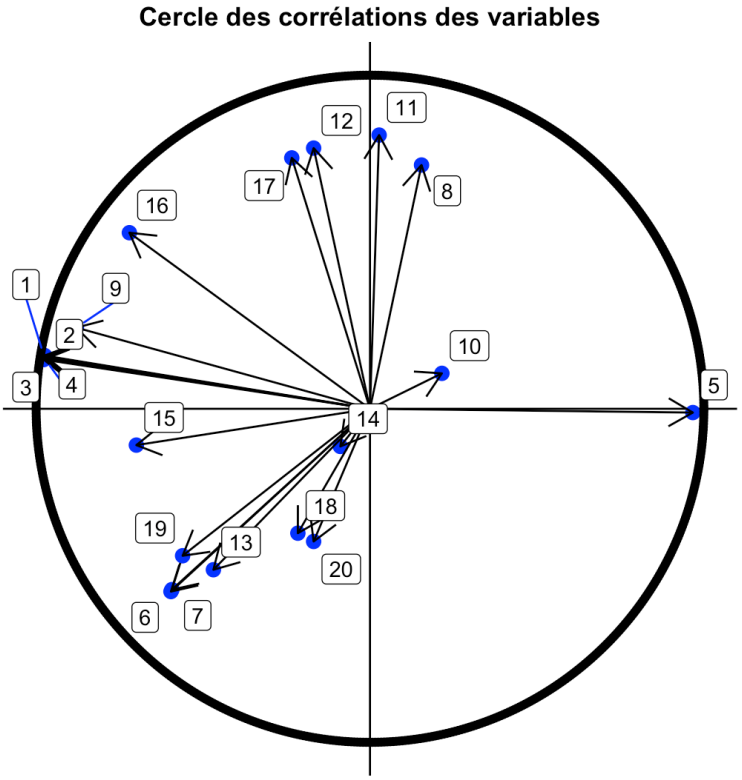
Présentation bidimensionnelle

A première vue à l'aide de ce tableau, on peut remarquer que certaines variables semblent se rapprocher. Leurs valeurs sont de même unité, leurs moyennes, médianes, valeurs min et max et 1er et 3eme écarts types sont pratiquement égaux. On peut facilement en déduire que ces variables présentent un type d'information proche. On peut même supposer que ces variables seront fortement positivement corrélées entre elles. Par exemple, les couples "Energie, règlement UE kJ/100g" et "Énergie, avec Fibres kJ/100g", "Energie, règlement UE kcal/100g" et "Énergie avec fibres kcal/100g" semblent similaires. Il en est de même pour "Protéines facteur de Jones" et "Protéines 6.25".

En revanche, les variables n'ont pas toutes les mêmes unités entre elles. On a à la fois des "kJ/100g", des "kcal/100g" et des "g/100g". Il est donc nécessaire de centrer et réduire les données afin d'effectuer une ACP normée afin de pouvoir correctement interpréter les résultats.

Représentation graphique

Cercle des corrélations des variables

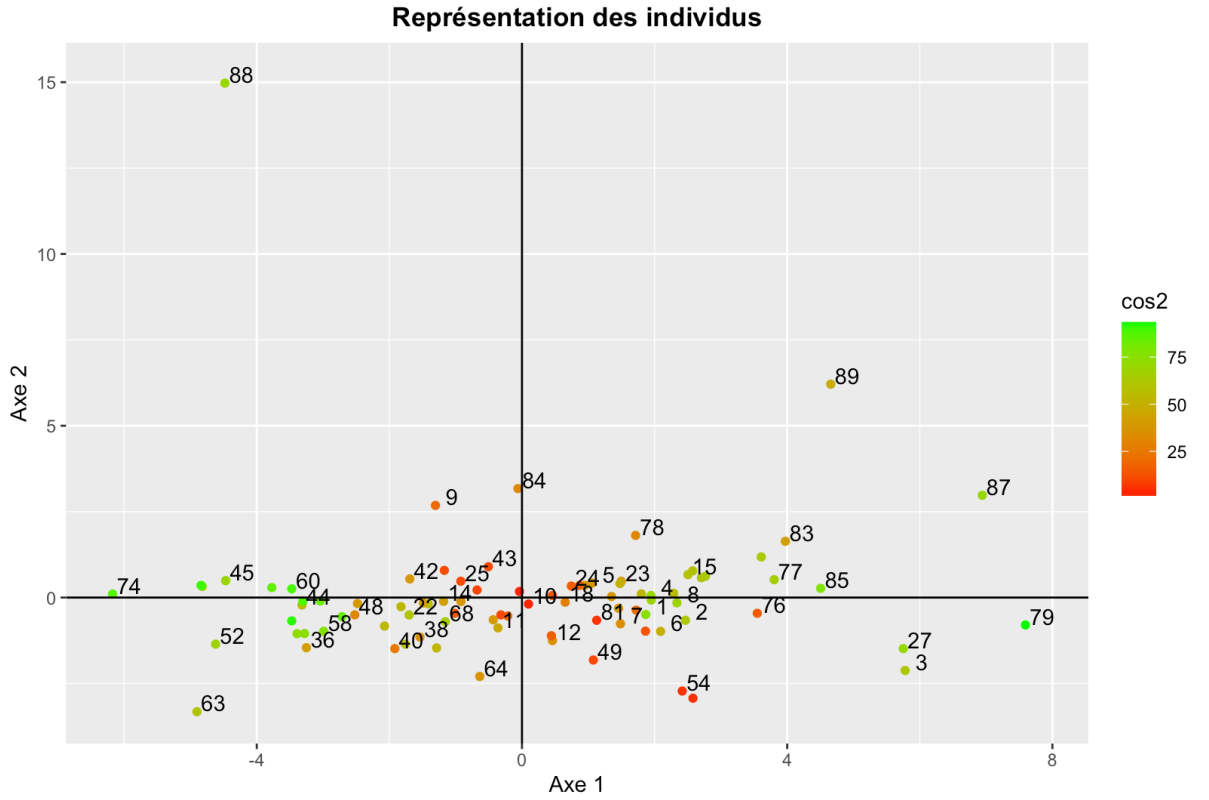


On peut remarquer que les variables 1, 2, 3 et 4 (que l'on regroupe et appellera "Énergie kj kcal") et la variable 5 "Eau" sont bien très représentées, elles sont fortement positivement corrélées. Les variables 14 "Acides organiques", 10 "Sucres", 18 "Sel chlorure de sodium" et 20 "Sodium" sont quant à elles mal représentées. Autrement dit, ces variables sont très éloignées des extrémités du cercle. On peut également dire que 17 "Acide gras", 12 "Polyols", 11 "Fibres alimentaires" sont plutôt bien représentés aussi. Il en est de même pour le calcium (19), les protéines (6,7).

Les variables "Energie kj kcal", 5 "Eau", 9 "Lipides" et 15 "AG saturés" ont le plus contribué à l'axe V1. "Eau" à contribué positivement, tandis que "Energie kj kcal", "Lipides" et "AG saturés" ont contribué négativement à l'axe, on peut en déduire que plus le produit contient de l'eau, moins il apportera d'énergie lors de la consommation. Ces variables ne semblent pas être corrélées avec les Fibres (11), les Polyols totaux (12), les AG polyinsaturés (17) et les glucides (8) qui, eux, ont le plus contribué positivement à l'axe V2.

Ainsi on a :
Fortement corrélés positivement : 1 avec 2 avec 3 avec 4 avec 9
Fortement corrélés négativement : (1 2 3 4 9) avec 5
Non corrélés : (1 2 3 4 9) avec (19 6 7)
(1 2 3 4 9) avec (17 12 11)
(17 12 11) avec (19 6 7)
(17 12 11) avec 5 et (17 12 11) avec 16

Représentation des individus



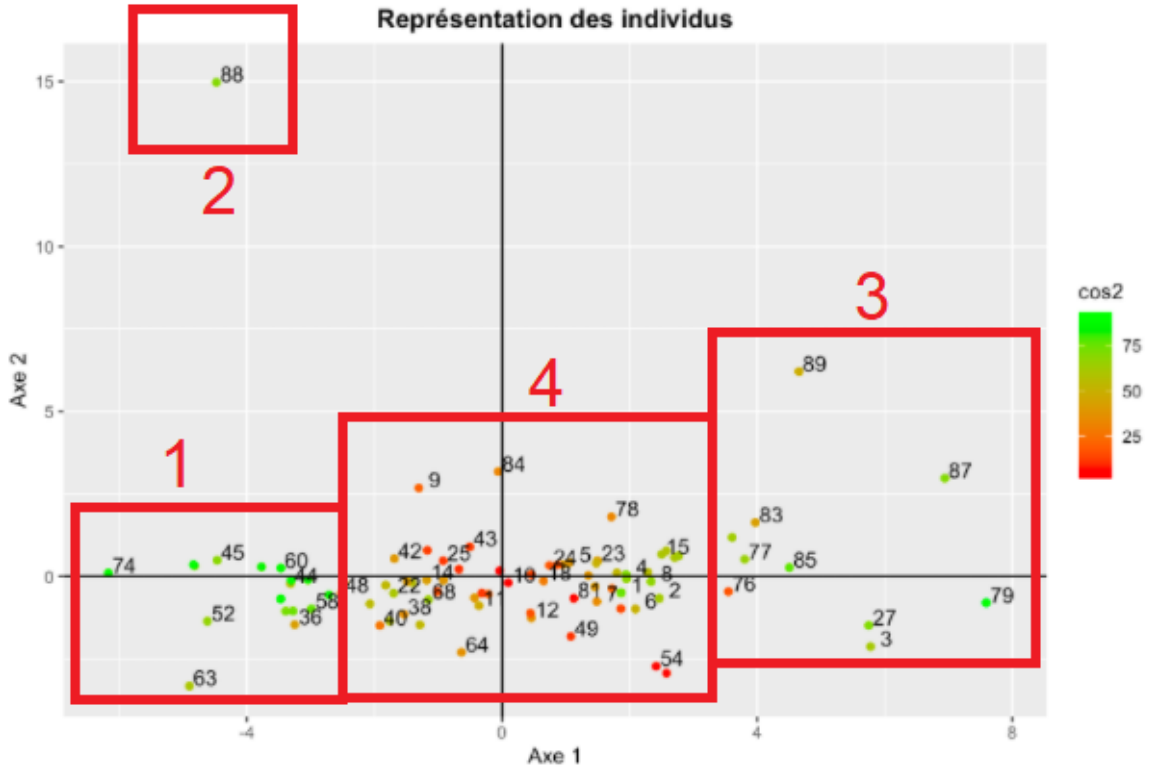
On peut remarquer que les individus 74, 88, 45, 60, 44, 52, 63, 15, 1, 4, 2, 8, 15, 77, 85, 27, 3, 79, 87 sont bien représentés. Les autres individus, quant à eux, se trouvent plus proches de l'origine du graphe et sont de mauvaise qualité de représentation. On peut donc en déduire qu'ils sont mieux représentés par d'autres axes que A1 et A2.

On remarque l'individu 88 qui se trouve seul très haut sur l'axe A2, on peut donc en déduire que ce fromage contient beaucoup d'acides gras et de fibres en comparaison aux autres individus.

On sait que les fromages à pâte pressée ou cuite ne contiennent que très peu d'eau à la fin de leur production. Au contraire, les fromages à pâte molle, à croûte fleurie, à pâte non cuite ou aux fromages frais contiennent, eux, beaucoup plus d'eau. Cela est cohérent avec les résultats de l'ACP, puisque l'on trouve des fromages tels que la cancoillote, la mozzarella, le fromage de chèvre ainsi que le fromage type camembert, autrement dit des fromages à pâte molle, croûte fleurie, dans la partie droite du plan. En revanche, des fromages tels que le fromage de brebis, la mimolette, le parmesan et le comté, des fromages cuits, pressés ou vieillis, se trouvent dans la partie gauche du plan.

On peut également remarquer qu'on a peu de fromage qui se démarquent par leur quantité en fibres, Polyols totaux, AG polyinsaturés et glucides.

Représentation des individus par classes



On peut rassembler les individus en quatre classes pour les catégoriser :

Le premier groupe rassemble les individus ayant une quantité d'eau très faible (individus 74, 45, 52, 60, etc.) et une quantité en lipide élevée. Il pourrait donc correspondre à des fromages plutôt secs. C'est par la même occasion des fromages qui devraient être très caloriques. Les résultats sont cohérents puisque l'on peut y retrouver le fromage 60 correspondant au cheddar qui est un des fromages les plus caloriques. Par la même occasion, on remarque le parmesan (52) et compté (45) qui sont des fromages réputés pour être secs.

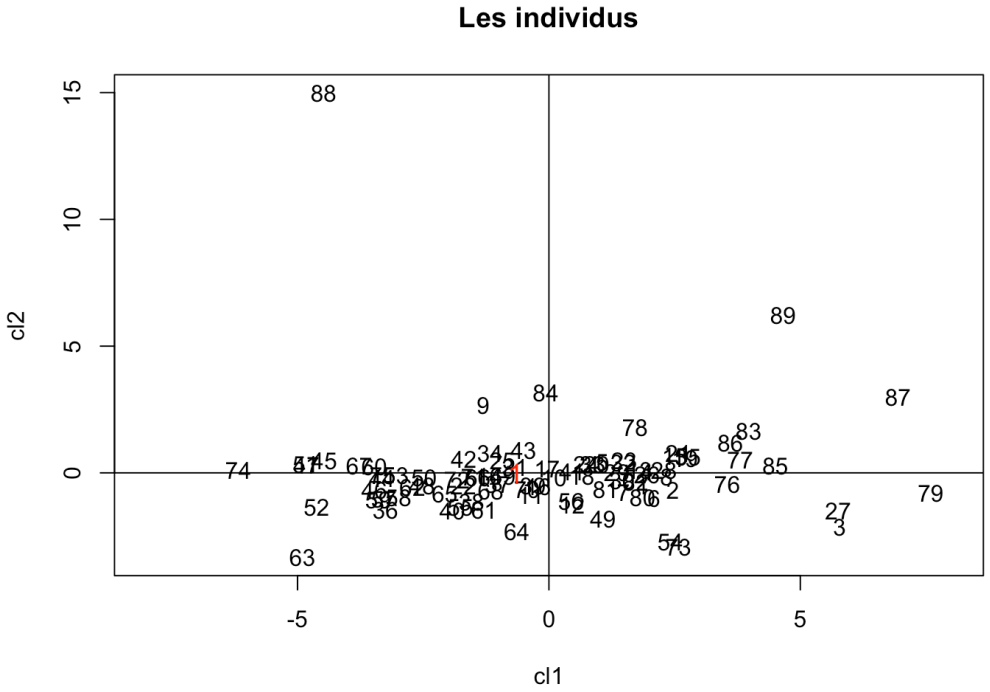
Le deuxième groupe rassemble un individu (88) et c'est celui ayant plus de fibres et d'acide gras que les autres. Ce résultat semble cohérent puisque l'individu 88 correspond à *"Spécialité végétale type fromage, à la noix de cajou, préemballée"* où la noix de cajou contient des fibres et est réputée pour être un aliment très riche en matière grasse.

Le troisième groupe, quant à lui, rassemble les individus ayant une quantité d'eau importante (individus 87, 79, 27, 3, 85, 77). Cette classe semble rassembler les fromages mou et les moins caloriques. On y trouve d'ailleurs le chèvre bûche et la mozzarella faisant partie des dix fromages les moins caloriques.

Le quatrième groupe rassemble, pour la plupart, des individus mal représentés dans ce plan. Cependant, quelques individus de ce plan semblent être bien représentés notamment l'individu 22,38,68,1,4,8,15,2. Ces individus correspondent à des fromages moyens qui ne se démarquent pas par une variable particulière. Par exemple, le camembert (individu 1 et 2) possède une quantité d'eau, de fibre, de glucide moyenne. Autrement dit, dans notre

ACP, il représente un fromage moyen. Ainsi lorsque l'on parle des fromages des autres classes on peut le comparer à celui là. Et ainsi dire, en prenant du recul sur ce que l'on annonce, que la mozzarella est plus calorique que le camembert, que le parmesan est plus sec que le camembert.

Individu supplémentaire



Le graphique ci-dessus représente les individus après avoir effectué l'ACP avec un individu supplémentaire, ici représenté en rouge. A première vue, ce graphique pourrait avoir aucune signification dans le projet. C'est lorsque l'on connaît le type de l'individu que ce graphique prend tout son sens. Effectivement, l'individu représenté en rouge correspond ici au fromage moyen de tout notre jeu de données. Or, on remarque que le point rouge est pratiquement représenté dans le centre de nos axes. L'ACP a pour but de rassembler dans le centre les individus moyens, banales et d'excentrer les autres. Ainsi, voir l'individu moyen parfaitement dans le centre confirme que l'ACP semble avoir du sens et être valide. Dans notre cas, cela est peu intéressant car le graphique est obtenu à partir d'ADE4 dont nous pouvons supposer que la librairie est fonctionnelle. En revanche, cela signifie aussi que le choix de nos données de départ est plutôt bon puisque l'ACP sur deux axes arrive à représenter une information valide.

Ainsi, à première vue ce graphique peut sembler non intéressant mais il est en réalité le témoin de la réussite de l'ACP sur notre jeu de données initial.

Résumé numérique

Variables

Contributions

Variables	V1	V2
Energie, Règlement UE N° 1169/2011 (kJ/100 g)	11.90	0.59
Energie, Règlement UE N° 1169/2011 (kcal/100 g)	11.86	0.61
Energie, N x facteur Jones, avec fibres (kJ/100 g)	11.93	0.52
Energie, N x facteur Jones, avec fibres (kcal/100 g)	11.89	0.55
Eau (g/100 g)	11.68	0.00
Protéines, N x facteur de Jones (g/100 g)	4.44	7.20
Protéines, N x 6.25 (g/100 g)	4.40	07.05
Glucides (g/100 g)	0.30	12.80
Lipides (g/100 g)	9.66	1.42
Sucres (g/100 g)	0.58	0.27
Fibres alimentaires (g/100 g)	0.01	16.14
Polyols totaux (g/100 g)	0.36	14.64
Cendres (g/100 g)	2.75	5.58
Acides organiques (g/100 g)	0.10	0.31
AG saturés (g/100 g)	6.12	0.28
AG monoinsaturés (g/100 g)	6.49	6.68
AG polyinsaturés (g/100 g)	0.69	13.56
Sel chlorure de sodium (g/100 g)	0.58	3.34
Calcium (mg/100 g)	3.93	4.65
Sodium (mg/100 g)	0.36	3.79

Les variables contribuant le plus à l'axe 1 sont les 4 variables “Energie kj/kcal" ainsi que les variables “eau” et “lipides”.
Les variables contribuant le plus à l’axe 2 sont “Fibres alimentaires”, “Polyols totaux”, “AG polyinsaturés” et "Glucides".
On peut donc en déduire la non corrélation de ces deux groupes de variables.

Qualités

Variables	V1	V2
Energie, Règlement UE N° 1169/2011 (kJ/100 g)	-95.26	2.45
Energie, Règlement UE N° 1169/2011 (kcal/100 g)	-94.99	2.55
Energie, N x facteur Jones, avec fibres (kJ/100 g)	-95.56	2.17
Energie, N x facteur Jones, avec fibres (kcal/100 g)	-95.19	2.31
Eau (g/100 g)	93.54	-0.01
Protéines, N x facteur de Jones (g/100 g)	-35.54	-30.03
Protéines, N x 6.25 (g/100 g)	-35.21	-29.39
Glucides (g/100 g)	2.38	53.41
Lipides (g/100 g)	-77.35	5.92
Sucres (g/100 g)	4.64	1.13
Fibres alimentaires (g/100 g)	0.08	67.33
Polyols totaux (g/100 g)	-2.85	61.08
Cendres (g/100 g)	-22.00	-23.29
Acides organiques (g/100 g)	-0.79	-1.27
AG saturés (g/100 g)	-48.97	-1.19
AG monoinsaturés (g/100 g)	-51.94	27.86
AG polyinsaturés (g/100 g)	-5.49	56.57
Sel chlorure de sodium (g/100 g)	-4.67	-13.91
Calcium (mg/100 g)	-31.50	-19.42
Sodium (mg/100 g)	-2.85	-15.83

On voit ici que les 5 premières variables sont de haute qualité (somme de leur colonne proche de 100). Ainsi, ces variables sont bien représentées dans le graphe. On remarque que le résultat obtenu est en parfaite adéquation avec le tableau des contributions précédent.

Individus

Contributions

Axe 1 :

Contribue le moins (-)	Contribue le plus (+)
Fromage à pâte molle et croûte lavée Reblochon Spécialité fromagère non affinée à tartiner	Cancoillotte Spécialité au soja Fromage de brebis Corse à pâte molle Fromage rond à pâte molle

Ces individus ont fortement contribué à la création de l'axe 1, ces fromages sont parmi ceux contenant le plus d'eau, on peut donc caractériser l'axe en échelle de quantité d'eau.

Axe 2:

Contribue le moins (-)	Contribue le plus (+)
Gorgonzola Fromage à pâte molle type Vieux Pané) Saint-Nectaire Munster	Spécialité au noix de cajou Spécialité en tranche Mimolette vieille Spécialité fromagère non affinée à tartiner

Ces individus sont les deux individus contenant le plus de fibres alimentaires, l'axe 2 peut être caractérisé par la quantité de fibres.

On peut remarquer que la contribution des individus aux axes est en parfaite adéquation avec les axes de notre cercle des corrélations.

Qualités

Axe 1 :

De moins bonne qualité (-)	De meilleure qualité (+)
Ossau-Iraty Cheddar Abondance Fromage de brebis à pâte pressée Fromage de brebis Corse à pâte molle Fromage de brebis des Pyrénées	Cancoillotte Camembert, sans précision Mozzarella au lait de vache Fromage de chèvre bûche, allégé en matière grasse Fromage rond à pâte molle et croûte fleurie

Axe 2 :

De moins bonne qualité (-)	De meilleure qualité (+)
Livarot Maroilles, sans précision Mimolette, sans précision Edam Fromage type Masdaam	Spécialité noix de cajou Spécialité à tartiner Spécialité en tranche Fromage à pâte molle triple crème Fromage fondu double crème

Les individus de meilleure qualité sont aussi ceux pour qui les axes sont proéminents dans leur données.

Interprétation des résultats

L'ACP présente des résultats cohérents, la classification des individus le long des axes concorde avec les données dans le dataset.

L'axe 1 représente la quantité d'eau dans un fromage, c'est une caractéristique importante de la classification des fromages. Elle dépend de la technique de fabrication et peut varier de manière importante entre les différents types de fromages. Le choix de cet axe se justifie facilement, et il offre des informations utiles sur les individus analysés.

On peut y voir une opposition entre les fromages contenant beaucoup d'eau aux fromages contenant une grande quantité d'énergie, mais aussi une opposition dans les techniques de fabrication du fromage. Les fromages à pâte pressée et cuite sont couramment plus secs et plus longtemps affinés, contrairement aux fromages au vieillissement rapide.

L'axe 2 quant à lui représente la quantité de Fibres alimentaires, les Polyols totaux, les AG polyinsaturés et les Glucides dans un fromage. Très peu de fromages en contiennent, mis à part les fromages au soja, ou les spécialités contenant un ingrédient supplémentaire (exemple : noix de cajou). Il y a donc une diversité très faible dans les individus, seuls un petit nombre de fromages sont opposés au reste des individus. Cela offre un potentiel d'interprétation faible, un autre choix d'axe aurait peut être été préférable. On peut expliquer cette observation par la variance cumulée pour deux axes qui était de 60.894%. Ainsi, nous représentons 61% de l'information en ayant choisi de réaliser l'ACP sur deux axes. Nous pouvons donc imaginer que l'axe 1 représente 50% de l'information et l'axe 2 10% au vu de la représentation des individus obtenues.

En réalité, choisir d'autres axes n'auraient pas été très utile car trois axes n'auraient ajouté que 12% d'information en plus et 22% de plus pour 4 axes. On aurait ainsi compliqué l'analyse des résultats pour une augmentation d'information pas suffisamment rentable.

Fonction R

Présentation fonction ACP

Notre fonction ACP prend en paramètre un dataframe contenant l'ensemble des données à analyser. Elle retourne une liste de cinq dataframes.

Paramètres d'entrée

Paramètre	Type	Description	Obligatoire	Défaut
dataval	dataframe	Dataframe contenant l'ensemble des données pour réaliser l'ACP	Oui	
scale	booléen	Réaliser une ACP non normée si le booléen est à FALSE	Non	TRUE

Valeurs de sortie

Valeur	Type	Description
datas	dataframe	Dataframe contenant l'ensemble des données pour réaliser l'ACP (données initiales)
indiv.coord	dataframe	Dataframe contenant l'ensemble des coordonnées des individus
indiv.cos2 *	dataframe	Dataframe contenant l'ensemble des qualités des individus
indiv.contr *	dataframe	Dataframe contenant l'ensemble des contributions des individus
var.coord	dataframe	Dataframe contenant l'ensemble des coordonnées des variables
var.cos2 *	dataframe	Dataframe contenant l'ensemble des qualités des variables
var.contr *	dataframe	Dataframe contenant l'ensemble des contributions des variables
vpi	dataframe	(**V**ecteurs **P**ropre **I**nertie) Dataframe contenant les vecteurs propres, les variances et variances cumulées

Les champs renseignés d'un astérisque sont ceux qui ne sont pas présents lorsque l'on effectue une ACP non normée.

Exécution

Notre fonction ACP commence par calculer le nombre de lignes et de colonnes du dataframe afin de pouvoir s'adapter à tout type de dataframe en entrée par la suite. Elle va ensuite centrer et, dans le cas d'une ACP normée, réduire la matrice. Ensuite, elle va calculer la matrice de variance-covariance. Nous nous sommes inspirés d'ADE4 pour créer un graphique des variances cumulées afin de proposer à l'utilisateur de choisir sur combien d'axes il voulait effectuer l'ACP. Par la suite, nous réalisons l'ACP en diagonalisant la matrice de variance-covariance. Nous poursuivons ensuite en calculant les axes principaux et les composantes principales. Ainsi, nous obtenons les coordonnées des individus. S'ensuit après l'obtention des coordonnées des variables. Par la suite, la fonction calcule la contribution et la qualité des variables et des individus. Pour finir, la fonction crée un dataframe pour chaque donnée importante et retourne l'ensemble de ses dataframes dans une liste.

Le code présent sur la **Figure E** explique progressivement le rôle de chaque ligne de code de la fonction. Enfin, nous avons souhaité rassembler l'ensemble des données importantes au sein d'un même objet sans avoir à appeler d'autres fonctions pour calculer des résultats supplémentaires. C'est pourquoi, l'objet retourné par la fonction ***acpFun*** peut paraître grand mais il contient toutes les informations importantes et utiles calculées lors de l'exécution du programme dont on souhaite faire bénéficier l'utilisateur qui l'utilise.

Présentation fonction affichage

Deux fonctions d'affichage ont été créées. Une première permet l'affichage des individus et l'autre du cercle des corrélations. Elles ont toutes les deux été créées à la main après diverses recherches sur internet. Ces fonctions utilisent principalement la librairie ggplot2.

Affichage des individus

La fonction s'intitule ***showIndivFun*** et prend en paramètre un dataframe, la valeur des axes x et y à représenter, un booléen label et number. Elle retourne un graphique ggplot représentant les individus sur les axes x et y.

Paramètre	Type	Description	Obligatoire	Par défaut
data	dataframe	Dataframe d'au minimum deux colonnes contenant les coordonnées des individus	Oui	
xvalue	integer	Numéro de la colonne dans le dataframe à afficher sur l'axe x	Non	1
yvalue	integer	Numéro de la colonne dans le dataframe à afficher sur l'axe y	Non	2
label	boolean	Affiche les individus sous forme d'étiquette	Non	TRUE
number	boolean	Affiche les individus par leur numéro de ligne à la place de leur nom	Non	TRUE
cos2	dataframe	Dataframe de même longueur que *data* contenant les qualités des individus	Non	NULL

Vous pourrez trouver sur la **Figure F** différents exemples d'appel de la fonction avec le résultat produit afin de mieux comprendre son fonctionnement.

Affichage des variables

La fonction s'intitule ***showCorrelFun*** et prend en paramètre un dataframe, la valeur des axes x et y à représenter, un booléen number. Elle retourne un graphique ggplot représentant le cercle des corrélations des variables.

Paramètre	Type	Description	Obligatoire	Par défaut
data	dataframe	Dataframe d'au minimum deux colonnes contenant les coordonnées des variables	Oui	
xvalue	integer	Numéro de la colonne dans le dataframe à afficher sur l'axe x	Non	1
yvalue	integer	Numéro de la colonne dans le dataframe à afficher sur l'axe y	Non	2
number	boolean	Affiche les variables par leur numéro de ligne à la place de leur nom	Non	TRUE

Vous pouvez voir des exemples d'affichage des individus par l'usage de cette fonction sur la **Figure G**.

Mise en oeuvre

ACP normée ou non normée ?

Tout d'abord, nous avons décidé de réaliser une ACP normée afin d'avoir un graphique beaucoup plus représentatif à la fin. De plus, toutes nos variables rentreront dans le cercle de corrélation et l'interprétation du résultat sera plus simple. De plus, si toutes nos données avaient été exprimées dans la même unité, le choix de normée ou non l'ACP aurait pu être discutable. Or, dans le cas précis de notre jeu de données des fromages, nous avons des valeurs dans des unités différentes comme expliqué ici.

Réalisation de l'ACP

Pour réaliser une ACP avec notre fonction, il faut simplement appeler la fonction en la mettant comme résultat d'une variable :

```
acpMan <- acpFun(fro)
```

Le paramètre passé à la fonction "**acpFun**" doit être un dataframe. Il s'agit du dataframe contenant l'ensemble de vos données à traiter.

Vous pouvez également effectuer une ACP non normée :

```
acpMan <- acpFun(fro,scale=FALSE)
```

Attention : Toutes les commandes ci-dessous ne s'appliquent pas si vous avez effectué une ACP non normée. (voir valeurs de sortie de la fonction ACP)

Ensuite, vous pouvez afficher les vecteurs propres, variances et variances cumulées en tapant la commande suivante :

```
show(acpMan$vpi)
```

Les commandes d'affichage ci-dessous présentent deux lignes pour les variables et quatre lignes pour les individus. Notez bien qu'il s'agit de différentes possibilités d'exécutions dont vous pourrez voir les exécutions sur la **Figure F** et la **Figure G**.

Variables

A présent, il est possible de gérer toute la partie variable en affichant tout d'abord le cercle des corrélations :

```
showCorrelFun(acpMan$var.coord)
showCorrelFun(acpMan$var.coord,number=FALSE)
```

Vous pouvez également afficher les coordonnées des variables sans utiliser d'affichage graphique. Pour se faire, tapez la commande suivante :

```
acpMan$var.coord
```

Pour afficher les contributions et la qualité des variables voici les commandes respectives :

```
round(acpMan$var.contr,2) #contribution variables
round(acpMan$var.cos2,2) #qualité variables
```

Individus

Vous pouvez afficher graphiquement tous les individus :

```
showIndivFun(acpMan$indiv.coord,label=FALSE,cos2=acpMan$indiv.cos2)
showIndivFun(acpMan$indiv.coord,label=FALSE)
showIndivFun(acpMan$indiv.coord,label=FALSE,number=FALSE,cos2=acpMan$indiv.cos2)
showIndivFun(acpMan$indiv.coord,cos2=acpMan$indiv.cos2)
```

Comme pour les variables, il est possible d'afficher seulement les coordonnées des individus pour se faire taper la commande suivante :

```
acpMan$indiv.coord
```

Pour afficher les contributions et la qualité des individus voici les commandes respectives :

```
round(acpMan$indiv.contr,2) #contribution des individus
round(acpMan$indiv.cos2,2) #qualité des individus
```

Comparaison avec ADE4

Fonctionnalité	Notre projet	ADE4
Valeurs propres		
Vecteurs propres		
Coordonnées des variables		
Contribution des variables		
Qualité des variables		
Coordonnées des individus		
Contribution des individus		
Qualité des individus		
Ajout d'un individu supplémentaire		
Ajout d'une variable supplémentaire		
ACP non normée		
Affichage cercle corrélations		
Représenter des classes de points		
Choix du nombre d'axe(s)		

Le tableau comparatif ci-dessus représente les principales fonctionnalités d'ADE4 et en aucun cas la totalité de celles-ci. Il permet de comparer quelles sont les fonctionnalités implémentées dans ADE4 et pas dans notre projet.

On constate que nous n'avons pas implémenté l'ajout d'un individu ou d'une variable supplémentaire. Nous avons souhaité nous concentrer sur des fonctionnalités que l'on trouvait plus importantes à implémenter. De plus, l'ajout de variables ou d'individus supplémentaires n'est pas quelque chose de difficile. Autrement dit, avec toutes les données de notre fonction ACP, nous sommes en mesure de calculer facilement les coordonnées d'une variable ou d'un individu supplémentaire.

L'autre fonctionnalité utile que nous n'avons pas implémentée est celle de la représentation des classes. Nous aurions aimé pouvoir ajouter une fonction qui calcule automatiquement des classes et les affichent avec un graphe ggplot2, mais nous avons manqué de temps. Contrairement à l'ajout d'individu ou de variable supplémentaire, rajouter cette fonctionnalité est beaucoup plus compliqué car elle demande plus de réflexion et de temps. La **Figure H** de l'annexe présente un exemple d'affichage que nous aurions aimé proposer pour le rendu du projet. Cependant, il faut noter que la classification visuelle est quelque chose qui se fait souvent facilement à la main contrairement au reste des fonctionnalités implémentées.

Comme vous pouvez le constater, la plupart des fonctionnalités d'ADE4 ont été implémentées. En revanche, notre projet semble tout de même posséder quelques avantages sur ADE4, les voici :

Fonctionnalité	**Notre projet**	**ADE4**
Retourne variance cumulée		
Affichage des individus par couleur selon la qualité		
Affichage des variables du cercle des corrélations sans superposition		

Il faut tout de même prendre du recul sur ce que nous avançons. En effet, nous ne connaissons pas l'intégralité des possibilités avec ADE4 et il se peut que le package soit composé de fonctionnalités dont nous ignorons l'existence. De plus, certaines fonctionnalités que nous proposons par rapport à ADE4 sont des fonctionnalités que le package aurait très bien pu ajouter, qui ne présentent aucune complexité à intégrer, mais qui ne sont pas là pour des questions de légèreté et de simplification d'usage. Par exemple, la variance cumulée peut se calculer à partir des résultats de l'ACP d'ADE4 mais nous l'avons affiché en tant que fonctionnalité supplémentaire car le package ne retourne pas cette valeur avec le calcul déjà fait contrairement à notre fonction.

Annexe

Figure A : Traitement des valeurs NA avec R

```
> ### TRAITEMENT DES DONNEES NA ###
> #Nombre de données NA
> sum(is.na(fromages))
[1] 2326
> #Nombre de données NA par colonne
> colSums(is.na(fromages))
                                Eau (g/100 g)
                                            8
      Protéines, N x facteur de Jones (g/100 g)
                                            1
                  Protéines, N x 6.25 (g/100 g)
                                            1
                            Glucides (g/100 g)
                                            1
                             Lipides (g/100 g)
                                            0
                              Sucres (g/100 g)
                                           18
                            Fructose (g/100 g)
                                           43
                           Galactose (g/100 g)
                                           81
                             Glucose (g/100 g)
                                           41
                             Lactose (g/100 g)
                                           35
```

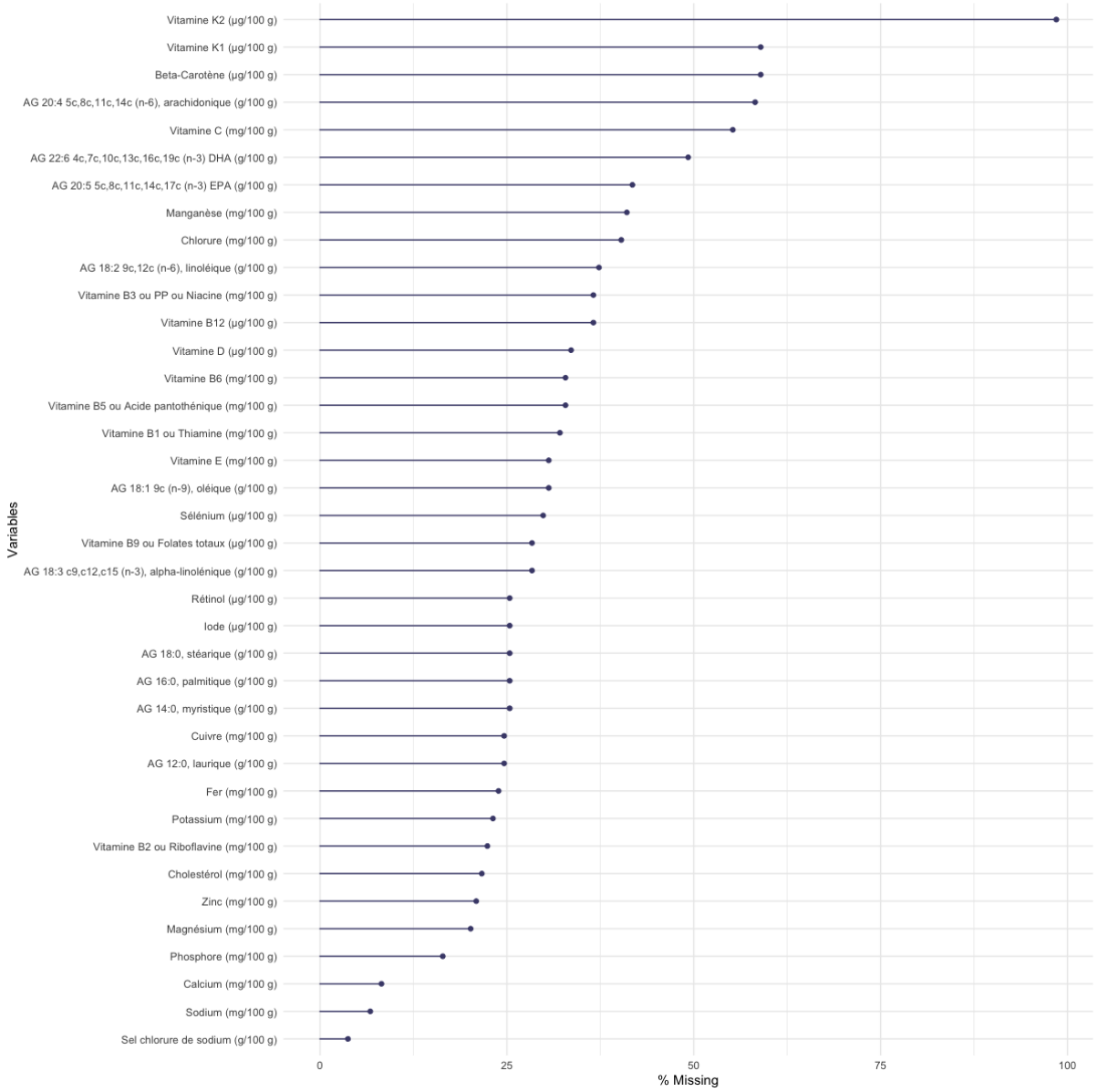
Extrait de l'exécution du code pour récupérer les valeurs NA des fromages

Figure B : Traitement des valeurs NA avec funModeling

```
> df_status(fromages)
                                          variable q_zeros p_zeros q_na   p_na q_inf p_inf      type unique
1                                     alim_grp_code       0    0.00    0   0.00     0     0 character      1
2                                    alim_ssgrp_code       0    0.00    0   0.00     0     0 character      1
3                                  alim_ssssgrp_code       0    0.00    0   0.00     0     0 character      6
4                                    alim_grp_nom_fr       0    0.00    0   0.00     0     0 character      1
5                                   alim_ssgrp_nom_fr       0    0.00    0   0.00     0     0 character      1
6                                 alim_ssssgrp_nom_fr       0    0.00    0   0.00     0     0 character      6
7                                           alim_code       0    0.00    0   0.00     0     0   numeric    134
8                                        alim_nom_fr       0    0.00    0   0.00     0     0 character    134
9                                       alim_nom_sci       0    0.00  134 100.00     0     0 character      0
10        Energie, Règlement UE N° 1169/2011 (kJ/100 g)       0    0.00   12   8.96     0     0   numeric     69
11      Energie, Règlement UE N° 1169/2011 (kcal/100 g)       0    0.00   12   8.96     0     0   numeric     92
12  Energie, N x facteur Jones, avec fibres  (kJ/100 g)       0    0.00   12   8.96     0     0   numeric     72
13 Energie, N x facteur Jones, avec fibres  (kcal/100 g)       0    0.00   12   8.96     0     0   numeric     93
14                                    Eau (g/100 g)       0    0.00    8   5.97     0     0   numeric    105
15            Protéines, N x facteur de Jones (g/100 g)       0    0.00    1   0.75     0     0   numeric     98
16                        Protéines, N x 6.25 (g/100 g)       0    0.00    1   0.75     0     0   numeric    100
17                                  Glucides (g/100 g)     108   80.60    1   0.75     0     0   numeric     24
18                                   Lipides (g/100 g)       0    0.00    0   0.00     0     0   numeric     95
19                                    Sucres (g/100 g)      63   47.01   18  13.43     0     0   numeric     27
20                                  Fructose (g/100 g)       3    2.24   43  32.09     0     0   numeric      8
21                                 Galactose (g/100 g)       2    1.49   81  60.45     0     0   numeric      9
22                                   Glucose (g/100 g)       4    2.99   41  30.60     0     0   numeric     11
23                                   Lactose (g/100 g)       3    2.24   35  26.12     0     0   numeric     25
```

Extrait de l'exécution du code de la fonction df_status de la libraire funModeling

Figure C : Traitement des valeurs NA avec naniar



Graphique obtenu après exécution de la fonction gg_miss_var de la librairie naniar sur le dataframe des fromages

Figure D : code de nettoyage du jeu de données initial

```
#installation du package ggrepel
install.packages("ggrepel")

#installation du package dplyr
install.packages("dplyr")

#installation du package funModeling
install.packages("funModeling")

#installation du package naniar
install.packages("naniar")

getFro <- function(){
#chargement de la librairie de lecture csv
library(readr)
#import des données
data_ciqual <- read_delim("Desktop/Master Informatique/Analyse des
données/Projet/data_ciqual.csv"
                          ,delim = ";", escape_double = FALSE, locale =
locale(encoding = "latin1"),
                          trim_ws = TRUE,show_col_types = FALSE)

#dimension du dataset
dim(data_ciqual)

#nom des colonnes
names(data_ciqual)

#compositions des colonnes, on remarque leur format caractère
str(data_ciqual)

# rien de vraiment intéressant car les colonnes sont des caractères
summary(data_ciqual)



###ISOLATION DES DONNEES FROMAGES###

#copie des données originales par sécurité
data <- data_ciqual

#classes de chaque colonne
sapply(data_ciqual, class)
```

```r
#nombre de lignes sur le fromage
length(data[data=='fromages et assimilés'])

#isolation des colonnes qui n'ont pas le bon format
i <- c(10: 76)

#changement du type des colonnes de caractère à numérique
# +
#remplacement des virgules par des points
data[ , i] <- apply(data_ciqual[ , i], 2,
                    function(x) as.numeric(sub(",", ".", as.character(x),
fixed = TRUE)))

#import de la librairie
library("dplyr")
#on garde seulement les fromages et assimilés
fromages <- dplyr::filter(data, grepl('fromages et assimilés',
alim_ssgrp_nom_fr))

#on sauvegarde l'individu supplémentaire
indiv_supp <- fromages[1,]

#suppression du fromage moyen
fromages <- fromages[-1,]
fromages
#sommaire des fromages
#bcp plus intéressant qu'avant car on a des données numériques pour chaque
colonne
summary(fromages)




### TRAITEMENT DES DONNEES NA ###
#Nombre de données NA
sum(is.na(fromages))

#Nombre de données NA par colonne
colSums(is.na(fromages))

library(funModeling)
df_status(fromages)
library(naniar)
```

```
missVar.p1 <- fromages[,1:38]
missVar.p2 <- fromages[,39:76]

gg_miss_var(missVar.p1,show_pct = TRUE)
gg_miss_var(missVar.p2,show_pct = TRUE)

#Nombre de données non renseignées
length(fromages[fromages=='-'])
#lors de la conversion des colonnes, les "-" ont été remplacé par des NA ce
qui explique
#qu'on obtient 0

#Elimination des colonnes "inutiles"
fromages <- fromages[,1:61]
indiv_supp <- indiv_supp[,1:61]
drops <- c("Sélénium (µg/100 g)","Potassium (mg/100 g)","Phosphore (mg/100
g)",
           "Manganèse (mg/100 g)","Magnésium (mg/100 g)","Iode (µg/100
g)","Fer (mg/100 g)",
           "Cuivre (mg/100 g)","Chlorure (mg/100 g)","Cholestérol (mg/100
g)",
           "AG 22:6 4c,7c,10c,13c,16c,19c (n-3) DHA (g/100 g)",
           "AG 20:5 5c,8c,11c,14c,17c (n-3) EPA (g/100 g)",
           "AG 20:4 5c,8c,11c,14c (n-6), arachidonique (g/100 g)",
           "AG 18:3 c9,c12,c15 (n-3), alpha-linolénique (g/100 g)",
           "AG 4:0, butyrique (g/100 g)",
           "AG 6:0, caproïque (g/100 g)",
           "AG 8:0, caprylique (g/100 g)",
           "AG 10:0, caprique (g/100 g)",
           "AG 12:0, laurique (g/100 g)",
           "AG 14:0, myristique (g/100 g)",
           "AG 16:0, palmitique (g/100 g)",
           "AG 18:0, stéarique (g/100 g)",
           "AG 18:1 9c (n-9), oléique (g/100 g)",
           "AG 18:2 9c,12c (n-6), linoléique (g/100 g)",
           "Fructose (g/100 g)",
           "Galactose (g/100 g)",
           "Glucose (g/100 g)",
           "Lactose (g/100 g)",
           "Maltose (g/100 g)",
           "Saccharose (g/100 g)",
           "Amidon (g/100 g)",
           "Alcool (g/100 g)",
           "alim_nom_sci"
           )
```

```
fromages <- fromages[ , !(names(fromages) %in% drops)]

indiv_supp <- indiv_supp[ ,!(names(indiv_supp) %in% drops)]

#observation des NA par colonne après le premier balayage
colSums(is.na(fromages))


#Elimination des lignes "inutiles"
fromages <- fromages[!is.na(fromages$`Energie, Règlement UE N° 1169/2011
(kJ/100 g)`),]
fromages <- fromages[!is.na(fromages$`Calcium (mg/100 g)`),]
fromages <- fromages[!is.na(fromages$`Sodium (mg/100 g)`),]
fromages <- fromages[!is.na(fromages$`AG saturés (g/100 g)`),]
fromages <- fromages[!is.na(fromages$`Eau (g/100 g)`),]
fromages <- fromages[!is.na(fromages$`AG polyinsaturés (g/100 g)`),]

#Dilemme entre suppression des lignes ou il n'y a pas de sucre renseigné
#ou des lignes ou il n'y a pas d'amidon renseigné
#Suppression des lignes où la valeur du sucre n'est pas renseigné
fromages <- fromages[!is.na(fromages$`Sucres (g/100 g)`),]

#observation des NA par colonne après le balayage par colonne puis par ligne
colSums(is.na(fromages))

#isolation des données nominatives
names <- as.data.frame(fromages[,8])
#isolation des données quantitatives
fro <- fromages[9:28]
indiv_supp <- indiv_supp[9:28]

rownames(fro) <- names[,1]
return(fro)
}
```

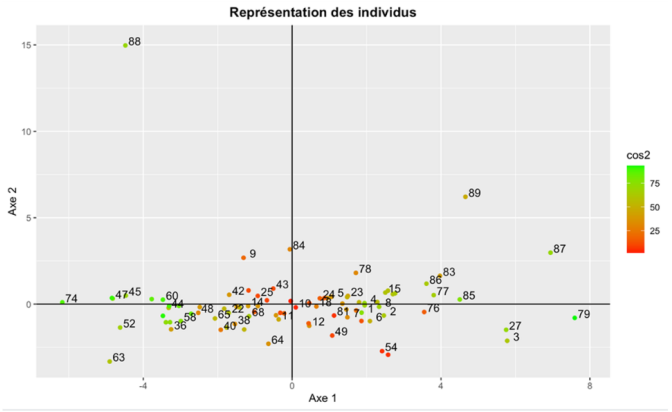
Figure E : Code de la fonction ACP

```r
acpFun <- function(dataVal,scale=TRUE){
  nblignes <- dim(dataVal)[1]
  nbcols <- dim(dataVal)[2]

  #centrée (réduite si normée)
  X <- scale(dataVal,center = TRUE, scale=scale)*sqrt(nblignes/(nblignes-1));

  #matrice de variance/covariance
  S = t(X)%*%X * (1/nblignes)

  #variance
  vari <- round(eigen(S)$values,3)
  #affichage sous forme de graphique
  barplot(main = "Variance",
          col="blue",
          vari)

  nb_axis <- readline(prompt="Combien d'axe ?")
  # conversion caractère en entier
  nb_axis <- as.integer(nb_axis)

  total <- sum(eigen(S)$values)
  #somme des valeurs des variances cumulées
  varCum <- c(round(cumsum(eigen(S)$values*(100/total)),3))

  # 4. ACP Normée diagonalisation de S
  ACP=eigen(S)

  # Inertie Ig
  inertie <- c(round(ACP$values,3))

  # axes principaux
  u=ACP$vectors

  # composantes principales F : individus
  F=X%*%u

  data <- as.data.frame(F[,1:nb_axis])

  #Contributions des individus
  indivCont <- F[,1:nb_axis]
  score <- indivCont^2
  for(i in 1:nb_axis){
   indivCont[,i] <- score[,i]/(nblignes*ACP$values[i])
  }
  indivCont<- as.data.frame(indivCont*100)

  #Qualité des individus
  indivQual <- F[,1:nb_axis]
  scoreL <- X^2
  scoreL <- rowSums(scoreL)
  for(i in 1:nb_axis){
    for(j in 1:nblignes){
```

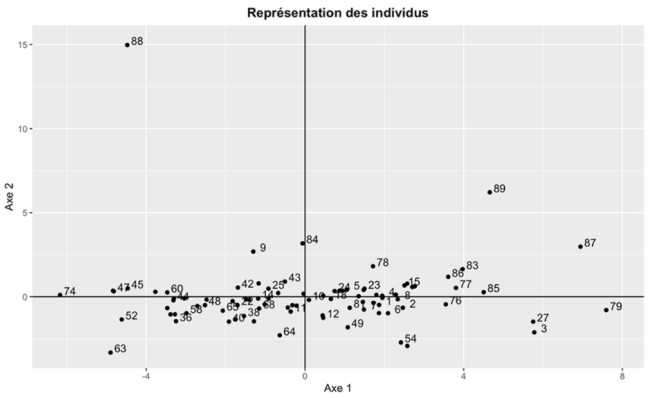
```
      indivQual[j,i] <- score[j,i]/scoreL[j]
    }
  }
  indivQual<- as.data.frame(indivQual*100*sign(F[,1:nb_axis]))

  # G : CP coord des variables
  G<-matrix(0,nbcols,nbcols)
  for(i in 1:nb_axis){
    G[,i]=sqrt(ACP$values[i])%*%ACP$vectors[,i]
  }

  #conversion matrice en dataframe
  dataCor=as.data.frame(G[,1:nb_axis])
  rownames(dataCor)<- colnames(dataVal)


  # CP qualité des variables
  dataCor.cos2 <- as.data.frame((G[,1:nb_axis]^2)*100*sign(G[,1:nb_axis]),row.names = colnames(dataVal))

  # CP contribution des variables
  dataCor.contr <- G[,1:nb_axis]^2
  for(i in 1:nb_axis){
    dataCor.contr[,i] <- dataCor.contr[,i] / ACP$values[i]
  }
  dataCor.contr <- as.data.frame(abs(dataCor.contr*100),row.names = colnames(dataVal))

  #Création de la matrice contenant l'inertie, la variance et la variance cumulée
  vpi <- matrix(c(inertie,vari,varCum),ncol = 3)
  colnames(vpi) <- c("Val. propre", "Variance (%)", "Variance cumul (%)")
  vpi <- as.data.frame(vpi)

  if(!scale){
    listOfDataframe = list(
      "datas" = dataVal,
      "indiv.coord" = data,
      "var.coord" = dataCor,
      "vpi" = vpi
    )
  }else{
    listOfDataframe = list(
      "datas" = dataVal,
      "indiv.coord" = data,
      "indiv.contr" = indivCont,
      "indiv.cos2" = indivQual,
      "var.coord" = dataCor,
      "var.cos2" = dataCor.cos2,
      "var.contr" = dataCor.contr,
      "vpi" = vpi
    )
  }
  return(listOfDataframe)
}
```

Figure F : Fonction affichage showIndivFun

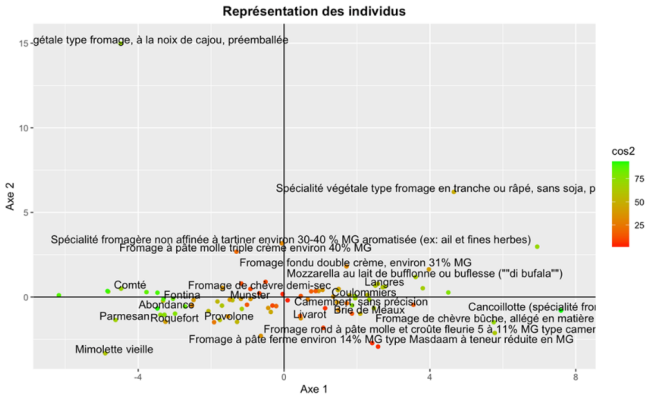
showIndivFun(acpMan$indiv.coord,label=FALSE,cos2=acpMan$indiv.cos2)



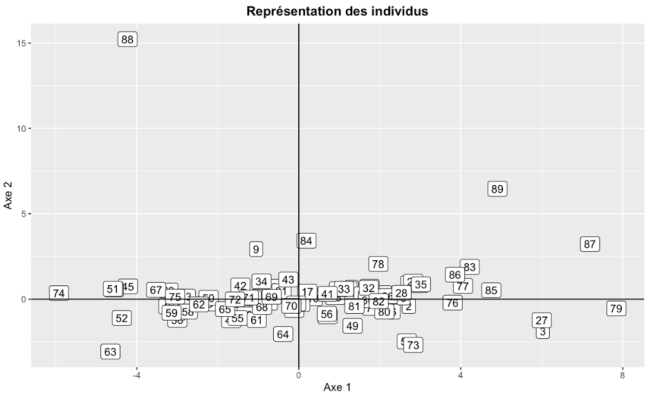
showIndivFun(acpMan$indiv.coord,label=FALSE)



showIndivFun(acpMan$indiv.coord,label=FALSE,number=FALSE,
cos2=acpMan$indiv.cos2)



showIndivFun(acpMan$indiv.coord,cos2=acpMan$indiv.cos2)

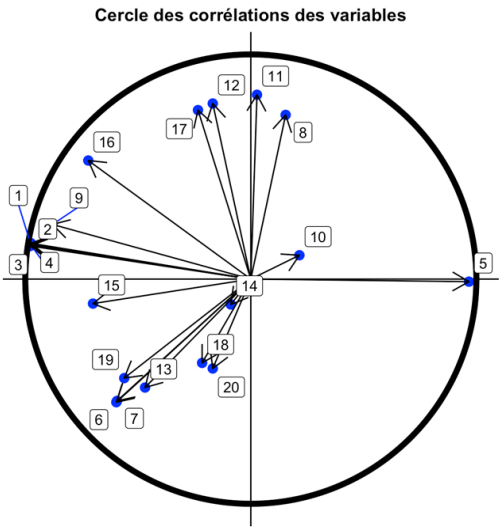


Graphique présentant les différentes possibilités de graphiques avec la fonction showIndivFun du projet selon les paramètres dans l'appel de la fonction

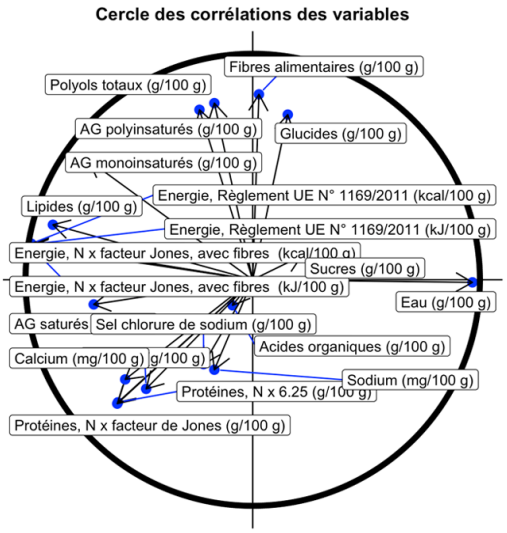
Figure G : Fonction affiche showCorrelFun

Graphique présentant les différentes possibilités de graphiques avec la fonction

showCorrelFun(acpMan$var.coord)

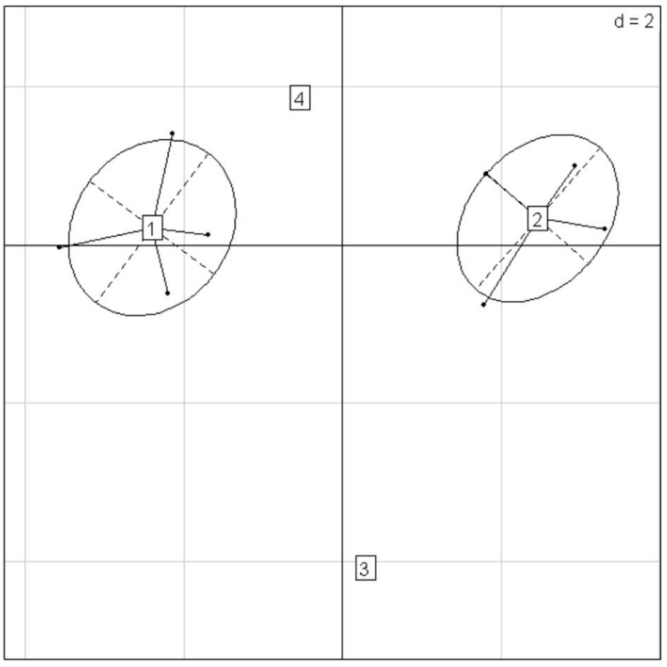


showCorrelFun(acpMan$var.coord,number=FALSE)



showCorrelFun du projet selon les paramètres dans l'appel de la fonction

Figure H : Exemple de représentation des classes avec ADE4



*Graphique représentant l'affichage des classes via la méthode "**s.class**" d'ADE4 (source : ©L. Bellanger)*

Figure I : Code de la fonction showIndivFun

```
showIndivFun <-
function(data,xvalue=1,yvalue=2,label=TRUE,number=TRUE,cos2=NULL){
  library(ggplot2)
  valueToShow <- c(1:dim(data)[1])
  if(!number){
    valueToShow <- rownames(data)
  }
  base <- ggplot(data, aes(x=data[,xvalue], y=data[,yvalue])) +
    ggtitle("Représentation des individus")+
    theme(plot.title = element_text(hjust = 0.5,face="bold"))+
    labs(y=paste(c("Axe", yvalue), collapse = " "), x =paste(c("Axe",
xvalue), collapse = " "))
  if(!is.null(cos2)){
    cos2 <- abs(cos2[,xvalue]+cos2[,yvalue])
    gradient<-base+geom_point(aes(color = cos2)) + # Show dots
      scale_color_gradient(low = "red", high = "green") #Update color
gradient
  }else{
    gradient<-base+geom_point()
  }
  if(label){
    base +
      geom_label(
        label=valueToShow,
        nudge_x = 0.25, nudge_y = 0.25,
        check_overlap = T
      )+ geom_hline(yintercept=0,color="black")+ geom_vline(xintercept =
0,color="black")
  }else{
      gradient+
      geom_text(
        label=valueToShow,
        nudge_x = 0.25, nudge_y = 0.25,
        check_overlap = T
      )+ geom_hline(yintercept=0,color="black")+ geom_vline(xintercept =
0,color="black")

    }
}
```

Figure J : Code de la fonction showCorrelFun

```
#Fonction trouvée sur internet
circleFun <- function(center = c(0,0),diameter = 1, npoints = 100){
  r = diameter / 2
  tt <- seq(0,2*pi,length.out = npoints)
  xx <- center[1] + r * cos(tt)
  yy <- center[2] + r * sin(tt)
  return(data.frame(x = xx, y = yy))
}

showCorrelFun <- function(data,xvalue=1,yvalue=2,number=TRUE){

  dat <- circleFun(c(0,0),2,npoints = 700)
  valueToShow <- c(1:dim(data)[1])
  if(!number){
    valueToShow <- rownames(data)
  }

  #affichage du cercle de corrélation
  library(ggplot2)
  library(ggrepel)

  variable.graphe <- ggplot(data, aes(data[,xvalue], data[,yvalue])) +
    ggtitle("Cercle des corrélations des variables")+
    theme(plot.title = element_text(hjust = 0.5,face="bold"))+
    geom_hline(yintercept=0,color="black")+
    geom_vline(xintercept = 0,color="black")+
    geom_point(color = "blue", size = 3)+
    geom_point(aes(x=x, y=y), data=dat,color="black")

  variable.graphe + geom_segment(aes(x = 0, y = 0, xend = data[,xvalue], yend
= data[,yvalue]),
                                 arrow = arrow(length = unit(0.5, "cm")))+
    geom_label_repel(aes(label = valueToShow),
                     box.padding   = 0.35,
                     point.padding = 0.5,
                     segment.color = 'blue') +
    theme(axis.line=element_blank(),axis.text.x=element_blank(),
          axis.text.y=element_blank(),axis.ticks=element_blank(),
          axis.title.x=element_blank(),
          axis.title.y=element_blank(),legend.position="none",
          panel.background=element_blank())+
    expand_limits(x=c(-1,1), y=c(-1, 1))+theme(aspect.ratio=1)
}
```

Figure K : Code de l'utilisation de la fonction ACP

```
### ACP ###
fro <- getFro()
acpMan <- acpFun(fro)

#Afficher les vecteurs propres, variances et variances cumulées
show(acpMan$vpi)

#Représentation des individus
showIndivFun(acpMan$indiv.coord,label=FALSE,cos2=acpMan$indiv.cos2)
showIndivFun(acpMan$indiv.coord,label=FALSE)
showIndivFun(acpMan$indiv.coord,label=FALSE,number=FALSE,cos2=acpMan$ind
iv.cos2)
showIndivFun(acpMan$indiv.coord,cos2=acpMan$indiv.cos2)

#affichage du cercle de corrélation
showCorrelFun(acpMan$var.coord)
showCorrelFun(acpMan$var.coord,number=FALSE)

#Contribution des variables
round(acpMan$var.contr,2)
#Qualité des variables
round(acpMan$var.cos2,2)

#Contribution des individus
round(acpMan$indiv.contr,2)
#Qualité des individus
round(acpMan$indiv.cos2,2)

#Possibilité d'effectuer une ACP non normée
acpMan <- acpFun(fro,scale=FALSE)
```

Figure L : Code de l'ACP avec ADE4

```
### VERSION AVEC ADE4 ###

library(ade4)
acp<-dudi.pca(fro,center=TRUE,scale=TRUE,scannf=TRUE)
round(acp$eig,2)
round(cumsum(acp$eig*5),2)

#ON ANALYSE LES VARIABLES
inertie <-inertia.dudi(acp, col.inertia=TRUE)
round(acp$co,2)

#contribution variable
round(inertie$col.abs,2)

#qualité de représentation des variables
round(inertie$col.rel,2)

#affichage du cercle des corrélations
s.corcircle(acp$co,xax=1,yax=2)

#Analyse des individus
ligsup<-suprow(acp,indiv_supp)

inertie <-inertia.dudi(acp, row.inertia=TRUE)
round(acp$li,2)

#Contribution individu
round(inertie$row.abs,2)

#Qualité de représentation
round(inertie$row.rel,2)

#Affichage graphique sans l'individu supplémentaire
s.label(acp$li,xax=1,yax=2)
#Affichage graphique avec l'individu supplémentaire
#coordonnées des individus actifs
cl1<-acp$li[,1]
cl2<-acp$li[,2]
#coordonnées des individus supplémentaires
csup1<-ligsup$lisup[,1]
csup2<-ligsup$lisup[,2]
#le graphique "vide"
plot(cl1,cl2,type="n",main="Les individus",xlim=c(-8,8))
```

```
abline(h=0,v=0)
#on ajoute les individus actifs
text(cl1,cl2,row.names(acp$li),)
#on ajoute les individus supplémentaires
text(csup1,csup2,row.names(ligsup$lisup),col="red",cex=1.2)

#isolement de l'individu supplémentaire
plot(cl1,cl2,type="n",main="Les individus",xlim=c(-8,8))
abline(h=0,v=0)
text(csup1,csup2,row.names(ligsup$lisup),col="red",cex=1.2)
```

Bibliographie

- https://www.laboitedufromager.com/les-differents-fromages-et-leurs-caracteristiques/
- http://www.math.u-bordeaux.fr/~mchave100p/wordpress/wp-content/uploads/2013/10/Exemple_interpret_ACP.pdf
- https://laboxfromage.fr/blog/fromages-les-moins-caloriques/
- https://laboxfromage.fr/blog/fromages-les-plus-caloriques/
- https://delladata.fr/visualisation-donnees-manquantes/
- https://ggplot2.tidyverse.org/

Lien du GitHub : https://github.com/thomaslprr/ACP