**Rapport du projet PCA**

Dans le cadre du projet de master 2 SAAD, mon objectif était de coder en python afin de réaliser une analyse en composante principale (PCA ou ACP) qui se rapproche le plus de ce qui se fait sur R avec le package FactoMineR. Sur la base du package existant en python fanalysis, j’ai amélioré ce dernier afin d’obtenir les fonctionnalités présentes sur R.

Le package fanalysis se compose de plusieurs modules pour les différentes méthodes d’analyse (ACP, ACM, ACS). Ici seul le module base et pca nous intéresse afin de réaliser nos objectifs. Ces modules définissent une classe qui comporte des méthodes afin de réaliser l’analyse souhaité.

La classe PCA possédait déjà plusieurs méthodes permettant de réaliser une ACP mais ces dernières n’étaient pas complètes, c’est pourquoi je me suis chargé d’améliorer la classe. De base, il était déjà possible d’obtenir les résultats d’une ACP basique, c’est-à-dire sans variables ou individus supplémentaires, ainsi que leur représentation de façon simple.

Par la suite, nous verrons le travail apporté afin de parfaire le package fanalysis. Je n’expliquerais pas tout ce qui était déjà présent dans le package.

**Ajout de variables qualitatives et quantitatives supplémentaires et d’individus supplémentaires**

Tout d’abord, il fallait se familiariser avec les classes PCA et Base afin de comprendre où il fallait intervenir pour ce que je souhaitais faire. Une fois cela effectué, un gros travail à faire était d’ajouter la possibilité d’avoir des variables qualitatives et/ou quantitatives supplémentaires ainsi que des individus supplémentaires. Afin de se rapprocher au plus de ce qui se fait sur R, j’ai codé pour rentrer en paramètre de la méthode fit, qui calcule les coordonnées des variables et individus, seulement les indices des variables et individus supplémentaire à l’instar du package FactoMineR et non les données ce qui nécessite un découpage des données de base par l’utilisateur. Ici, le découpage se fait directement dans la méthode fit.

Ligne 161 à 261 de pcabis.py :

On peut voir toutes les boucles if servant à découper le jeu de données de base suivant les variables et individus supplémentaire afin d’obtenir les données pour chaque catégorie, celle pour l’ACP et les autres pour tout ce qui est supplémentaire. C’est aussi ici que l’on récupère le nom des variables s’il y en a afin de les retranscrire par la suite dans la partie visualisation de l’utilisateur.

Ligne 380 à 422 de pcabis.py :

Une fois toutes les données bien séparées il faut calculer les coordonnées pour tout ce qui est supplémentaire étant donné que les celles pour les variables et individus de base étaient déjà dans le code. On calcule les coordonnées pour les individus supplémentaires, les variables quantitatives supplémentaires ainsi que les variables qualitatives supplémentaires.

Ligne 428 à 456 de pcabis.py :

De la même manière de ce qui avait déjà était fait pour les variables et individus, on code le label des individus et variables supplémentaire s’il n’en avait pas dans le jeu de données de base.

Ligne 502 à 528 de pcabis.py :

Par la suite on calcule le cos² des variables et individus supplémentaires dans une méthode \_compute\_stats qui n’intervient seulement si stats = True dans l’initialisation ce qui est le cas par défaut. On calcule donc le cos² pour chaque variable et individus supplémentaire selon les formules de l’ACP.

Ligne 539 à 545 de pcabis.py :

On calcule la corrélation entre les variables supplémentaires et les axes.

**Amélioration de la visualisation des statistiques sous forme de table**

Une fois toutes les statistiques calculées, nous pouvons nous attarder sur la visualisation de ces dernières. Pour ce faire, plusieurs améliorations et ajouts ont été apporté au package.

Ligne 723 à 930 de pcabis.py :

Toutes ces méthodes créées permettent une meilleure visualisation des vecteurs et valeurs propres, des variables et individus supplémentaires ou non sous forme de table stylisés et non brut comme c’était le cas. Cependant, cela est purement visuel car l’accès aux valeurs se fait toujours de manière simple par les attributs de la classe PCA. Toutes ces méthodes show\_xxxx permettent de créer la méthode summary affichant presque toutes les mêmes informations que la fonction summary du package FactoMineR. Il manque seulement quelques statistiques comme Dist ou v-test qui ne sont pas essentielles pour l’interprétation des résultats d’une ACP.

**Amélioration du graphique des valeurs propres et des contributions**

Des petites améliorations des graphiques des valeurs propres et des contributions sont également apportées.

Ligne 340 à 343, 350 à 354, 361 à 365 de basebis.py :

Pour le graphique des valeurs propres basique, une ligne horizontale est ajoutée pour y = 1 afin de représenter la règle de Kaiser.

Pour tous les graphiques des valeurs propres, on ajoute également la valeur de ces valeurs propres que ce soit en pourcentage ou autre ainsi qu’une ligne reliant les barres des valeurs propres afin de bien visualiser lesquelles sont à gardées.

Ligne 571 et 632 de basebis.py :

On ajoute aux graphiques des contributions une ligne rouge pointillée permettant de visualiser les contributions significatives.

**Ajout de la représentation des variables quantitatives supplémentaires au cercle des corrélations**

Ligne 590 à 608 de pcabis.py :

On ajoute les variables quantitatives supplémentaires au cercle des corrélations en rouge.

**Ajouts et améliorations de la représentation graphiques des individus**

Ligne 694 à 708 de pcabis.py :

On ajoute la représentation des individus supplémentaires ainsi que des variables qualitatives supplémentaires au graphiques des individus.

Ligne 649 à 674 de pcabis.py :

On ajoute la possibilité de représenter le point des individus du graphique selon leur cos² ou leur contribution par rapport au premier axe rentré en paramètre de la méthode mapping\_row mais également selon la variable qualitative supplémentaire, cependant je pense que cela ne marche pas s’il y a plusieurs variables qualitatives supplémentaires.