TD_01_Analyse : l'Analyse en Composante Principale (ACP)

Septembre 2025

1 Rendu (pour l'ensemble des TDs d'analyse)

Vous remettrez sur moodle pour le **5 Octobre** : un rapport au format pdf de vos analyses, un fichier python qui permet de générer vos graphiques, le fichier csv que vous utilisez.
Le rapport au format pdf inclus :

- une introduction
- une présentation de votre jeu de donnée
- une technique d'analyse de données **quantitatives : l'ACP**
- une technique d'analyse de données **catégorielles/qualitatives**
- une conclusion qui résume l'ensemble de vos analyses

2 Prérequis

Les librairies dont vous aurez besoins :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Lecture de votre fichier CSV

```
data = pd.read_csv()
```

Pensez à nettoyer votre jeu de donnée avant de procéder à l'analyse. L'ACP est une méthode d'analyse de données quantitatives. Vous prendrez soins de sélectionner des données appropriées pour ce TD.

3 Analyse en composante principale

Vous commencerez par standardiser l'ensemble des variables. La standardisation (aussi appelée normalisation standard) consiste à soustraire la moyenne et à la diviser par l'écart type. La distribution résultante a une moyenne de 0 et un écart-type de 1.

$$\text{valeur standardisée} = \frac{\text{valeur - moyenne}}{\text{écart-type}}$$

En deux lignes en python :

```
temp = x.sub(x.mean())
x_scaled= temp.div(x.std())
```

Nous appelons *x_scaled* le jeu de valeur standardisées au quel nous appliquerons l'ACP. Dans l'exemple ci-dessous le nombre de composantes conservées est fixé à 3, vous changerez cette valeur en fonction de vos besoins.

```
pca = PCA(n_components=3)
pca.fit(x_scaled)
```

pca.fit permet d'obtenir la modélisation de l'ACP. Afin d'obtenir les résultats pour les individus il faut combiner les fonctions $fit()$ et $transform$ de l'ACP. Une autre possibilité est d'utiliser la fonction $fit\_transform()$ qui réalise automatiquement la combinaison des deux.

```
pca_res= pca.fit_transform(x_scaled)
```

Nous appelons par la suite *pca_res* les résultat obtenus pour les individus.

3.1 Calcule des valeurs propres

Parmi l'ensemble des valeurs de la PCA vous retrouverez :

- Les valeurs propres des composantes : *pca.singular_values_*
- Les pourcentages de valeurs propres : *pca.explained_variance_ratio_*

Avec ces éléments créé une table qui résume les valeurs propre. Ce tableau inclura :

- Les dimensions de votre ACP
- Les valeurs propre
- Les pourcentages de valeur propre
- Les pourcentage de valeur propre cumulés

```
eig = pd.DataFrame({
    "Dimension" :
     ["Dim" + str(x + 1) for x in range(6)],
    "Valeur propre" : pca.explained_variance_,
    "% valeur propre" :
     np.round(pca.explained_variance_ratio_*100),
    "% cum. val. prop." :
     np.round(np.cumsum(pca.explained_variance_ratio_)*100)
    })
```

Une autre façon d'analyser les résultats des valeurs propre est de réaliser un diagramme bâton. Réalisez ce diagramme dont l'abscisse sera composé de l'ensemble des dimensions de votre ACP et l'ordonnée les pourcentages de valeur propre.

```
y1 = list(pca.explained_variance_ratio_)
x1 = range(len(y1))
plt.bar(x1,y1)
plt.show()
```

3.2 Graphique des variables

Pour afficher le graphique des variables vous pouvez utiliser la fonction biplot[1] dans le fichier python ci-joint. Voici un exemple ci-dessous.

```
biplot(score=data_sortie[:,0:2],
    coeff=np.transpose(pca.components_[0:2, :]),
    cat=y1[0:1], density=False)
plt.show()
```

Il n'y as pas le nom des variables sur ce graphique, faites les apparaître grâce à $coeff\_labels = list(x.columns)$.

3.3 Graphique des individus

Pour cette partie vous chercherez à afficher le graphique des individus de l'ACP. Pour ce faire vous pouvez commencer par créer un dataframe qui inclus les résultats des deux premières dimensions de l'ACP pour l'ensemble de vos individus. Vous ajouterez également à ce dataframe des variable catégorielles.

Dans un premier temps générez le graphique des individus de l'ACP sans distinction entre les individus. Pour ce faire vous pouvez faire un dataframe qui inclue vos deux premières dimensions d'ACP. Vous pouvez également ajouter des données catégorielles qui nous servirons par la suite pour colorier les individus dans une représentation graphique plus avancée. Dans l'exemple ci-dessous "goodanswer" est une donnée catégorielle.

[1] https://sites.google.com/view/aide-python/statistiques/machine-learning-en-python/analyses-en-composantes-principales?authuser=0#h.w5770aq0hr6z

```
pca_df = pd.DataFrame({
    "Dim1" : data_sortie[:,0],
    "Dim2" : data_sortie[:,1],
    "goodanswer" : df["goodanswer"]
})

pca_df.plot.scatter("Dim1", "Dim2")
plt.xlabel("Dimension 1 (%)")
plt.ylabel("Dimension 2 ( %)")
plt.suptitle("Premier plan factoriel (%)")
plt.show()
```

Dans un second temps vous générerez un graphique en colorant les individus d'après une de vos variables catégorielles. Dans l'exemple suivant j'utilise ma variable catégorielle "goodanswer". Vous commencerez par créer une palette de couleur basée sur Color Brewer.

```
palette = plt.get_cmap("Dark2")
couleurs = dict(zip(pca_df["goodanswer"].drop_duplicates(),
    palette(range(10))))
position = dict(zip(couleurs.keys(), range(10)))
```

Ensuite vous afficherez le graphique des individus en coloriant ces dernier grâce à la palette de couleur comme dans l'exemple ci-dessous. Je rappel que *good_answer* est dans l'exemple suivant la variable catégorielle utilisée pour la coloration.

```
pca_df.plot.scatter("Dim1", "Dim2",
    c = [couleurs[p] for p in pca_df["goodanswer"]])
for cont, coul in couleurs.items():
plt.scatter(3, position[cont] / 3 + 2.15, c = [coul], s = 20)
plt.text(3.2, position[cont] / 3 + 2, cont)
plt.xlabel("Dimension 1 (%)")
plt.ylabel("Dimension 2 ( %)")
plt.suptitle("Premier plan factoriel (%)")
plt.show()
```