TD_02_Analyse : l'Analyse Factorielle des Correspondances (AFC) et Analyse de Correspondances Multiples (ACM)

Septembre 2025

1 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'AFC et l'ACM sont des méthodes d'analyse de données qualitatives (catégorielles). Vous prendrez soins de sélectionner des données appropriées pour ce TD.

Les librairies nécessaires :

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

2 Analyse Factorielle des Correspondance

Afin de réaliser l'analyse factorielle des correspondance vous commencerez par réaliser un tableau de contingence entre les deux variables que vous souhaitez analyser. Pour ce faire vous pouvez utiliser la fonction 'crosstab' de la librairie panda. Voici un exemple :

```
data_crosstab = pd.crosstab(x['certitude'], x['goodanswer'])
```

Une fois cette étape réalisée vous procéderez à la standardisation des données (cf. TD_01_Analyse_ACP).

```
temp = data_crosstab.sub(data_crosstab.mean())
data_scaled= temp.div(data_crosstab.std())
```

Nous allons maintenant nous assurer qu'il est pertinent de réaliser une AFC sur ces données. Pour ce faire nous faisons le test suivant :test de sphéricité de Bartlett (*Bartlett Sphericity Test* en anglais). Ce test vérifie les intercorrelations entre les variables en faisant la comparaison entre la matrice de corrélation et la matrice identité. Si les variables sont indépendantes, la matrice de corrélation est égale à la matrice identité.

Afin de pouvoir réaliser une AFC, il est nécessaire que les variables soient globalement dépendantes, ie. que la matrice de corrélation et la matrice d'identité ne soient pas les mêmes. Il faut pour cela que la p-value du test de sphéricité de Bartlett soit le plus proche possible de 0.

```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value, p_value=calculate_bartlett_sphericity(data_scaled)
print(p_value)
```

A partir du moment où vos données sont valides pour une AFC, vous pouvez maintenant déterminer le nombre de facteurs à conserver pour votre analyse. Les facteurs, les variables latentes, sont des combinaisons linéaires des points modalités (lignes ou colonnes) exprimés par des profils (lignes ou colonnes). Le nombre maximal de facteurs est égal au minimum du nombre de lignes et de colonnes -1 :

$$n\_factors_{max} = min(col - 1, li - 1)$$

Nous commencerons par réaliser une première analyse sans rotation afin de déterminer le nombre de facteurs à fixer pour l'analyse.

```
fa = FactorAnalyzer(n_factors=6, rotation=None)
fa.fit(data_scaled)
ev, v = fa.get_eigenvalues()
print(ev)
```

Nous afficherons les résultats grâce à un graphique.

```
plt.scatter(range(1,data_scaled.shape[1]+1),ev)
plt.plot(range(1,data_scaled.shape[1]+1),ev)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigenvalue')
plt.grid()
plt.show()
```

Seuls les facteurs avec une valeur propre supérieur ou égale à 1 seront gardés. Nous pouvons finalement réaliser l'AFC avec le nombre de facteurs appropriés. Le code ci-dessous permet d'afficher les AFC en fonctions de 3 rotations différentes, les représentation graphiques seront donc légérement différentes. (FA pour *Factorial Analisys*).

```
methods = [
    ("FA No rotation", FactorAnalysis(2,)),
    ("FA Varimax", FactorAnalysis(2, rotation="varimax")),
    ("FA Quartimax", FactorAnalysis(2, rotation="quartimax")),
]
fig, axes = plt.subplots(ncols=3, figsize=(10, 8), sharex=True, sharey=True)

for ax, (method, fa) in zip(axes, methods):
    fa = fa.fit(data_scaled)

    components = fa.components_

    vmax = np.abs(components).max()
    ax.scatter(components[0,:], components[1, :])
    ax.axhline(0, -1, 1, color='k')
    ax.axvline(0, -1, 1, color='k')
    for i,j, z in zip(components[0, :], components[1, :], data_scaled.columns):
        ax.text(i+.02, j+.02, str(z), ha="center")
    for i,j, z in zip(components[0, :], components[1, :], data_scaled.index):
        ax.text(i+.02, j+.02, str(z), ha="center")
    ax.set_title(str(method))
    if ax.get_subplotspec().is_first_col():
        ax.set_ylabel("Factor 1")
    ax.set_xlabel("Factor 2")

plt.tight_layout()
plt.show()
```

3 Analyse des correspondances multiples (ACM)

Afin de procéder à cette analyse un pré-traitement sur les variables est de nouveau requis. Commencez par sélectionner l'ensemble des variables qualitatives de votre choix puis transformez votre dataframe en un tableau disjonctif complet.

Par exemple :

Race	Classe	Sexe
Elfe	Mage	F
Humain	Alchimiste	H
Tieflin	Alchimiste	NB

Transformé en tableau disjonctif :

Race	Classe Mage	Classe Alchimiste	Sexe F	Sexe H	Sexe NB
Elfe	1	0	1	0	0
Humain	0	1	0	1	0
Tieflin	0	1	0	0	1

Pour ce faire vous utiliserez la fonction get_dummies de la librairie panda.

```
dc=pd.DataFrame(pd.get_dummies(x[["certitude","goodanswer","img"]]))
dc.head()

from mca import MCA
mcaFic=MCA(dc,benzecri=False)

plt.scatter(mcaFic.fs_c()[:, 0],mcaFic.fs_c()[:, 1])
for i, j, nom in zip(mcaFic.fs_c()[:, 0],mcaFic.fs_c()[:, 1], dc.columns):
        plt.text(i, j, nom)
plt.show()
```